НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
"___"_____2020 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційні технології моніторингу довкілля»

спеціальності 122 «Комп'ютерні науки та інформаційні технології»

на тему: «Система моніторингу стану пріоритетів вступників до КПІ ім. Ігоря Сікорського»

Виконав: студент IV курсу, групи ТМ-61

Мовчан Владислав Олександрович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник старший викладач, Мірошниченко І.В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 122 Комп’ютерні науки та інформаційні технології

Спеціалізація Інформаційні технології моніторингу довкілля

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В. _____
(прізвище, ініціали) (підпис)

«____» _____ 2020р.

З А В Д А Н Н Я
НА ДИПЛОМНУ РОБОТУ СТУДЕНТУ

Мовчану Владиславу Олександровичу
(прізвище, ім’я, по батькові)

1. Тема дисертації Система моніторингу стану пріоритетів вступників до КПІ ім. Ігоря Сікорського

Науковий керівник Мірошниченко Іван Володимирович, старший викладач
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від “___”_____20___року №___

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи: Персональний комп’ютер під керуванням операційної системи Windows 10 Pro, мови програмування Javascript, середовище розробки WebStrom

4. Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) аналіз існуючих алгоритмів систем моніторингу, вибір методів та інструментів для розробки системи, розробка архітектури системи, вибір та реалізація засобів розробки програмного забезпечення, розробка веб-додатку з інтеграцією системи

5. Орієнтований перелік ілюстративного матеріалу схема роботи системи моніторингу, комплексна архітектура програмного продукту, схема MVC/Redax/React, діаграма послідовності

6. Консультанти розділів

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «____»_____201___р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Аналіз проблеми створення систем моніторингу	10.09.2018-9.11.2018	
2	Аналіз існуючих систем моніторингу	10.11.2018-11.03.2019	
3	Огляд та аналіз методів сортування та асинхронних запитів	12.03.2019-28.06.2019	
4	Аналіз алгоритмів систем моніторингу	29.06.2019-09.08.2019	
5	Аналіз вимог до розробки системи	10.08.2019-20.09.2019	
6	Моделювання роботи системи	20.09.2019-17.01.2020	
7	Розробка архітектури системи	18.01.2020-14.02.2020	
8	Розробка користувацького інтерфейсу	15.02.2020-28.02.2020	
9	Розробка системи	29.02.2020-27.03.2020	
10	Оформлення документації	28.03.2020-30.04.2020	

Студент . _____ ____Мовчан В.О.____
(підпис) (прізвище та ініціали)

Науковий керівник _____ Мірошниченко І.В.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Мета роботи - аналіз існуючих систем моніторингу, розробка нової системи моніторингу стану пріоритетів вступників до КПІ, яка буде в режимі реально часу показувати всім охочим детальну інформацію про вступ конкретного студента відсортовану по пріорітетам, розробка веб-додатку, який буде мати зручний інтерфейс та демонструватиме роботу системи.

Для вирішення поставленої задачі було сформовано перелік завдань, які визначили структуру дослідження:

— Проаналізувати предметну область, визначити інструменти для програмної реалізації
— Створення зручного та мінімалістичного UI інтерфейсу
— Створення мокової бази даних для прикладу моніторингу у форматі JSON
— Розробка логіки для сортування по пріоритетності та живого пошуку
— Розробка роутингу для пошуку з перезавантаженням сторінки
— Тестування роботи програми та пошук багів
— Усунення багів та удосконалення конкретних частин інтерфейсу виведення таблиць, знайдених в ході тестування

Для розв’язання поставлених задач, використовувались мови програмування HTML/CSS, Javascript, також використовувались алгоритми для сортування студентів по пріоритетності. Для мокової бази даних було додано асинхронних запит на сервер за допомогою бібліотеки Redux-Saga та імітація самої бд за допомогою бібліотеки Redux у форматі JSON. Система була написана на фреймворці React. Розробка системуздійснюється в середовищі WebStorm 2020.

Записка містить 79 сторінок, 25 рисунків, 2 додатки і 15 посилань.

Ключові слова: вступна компанія, система моніторингу, MVC, React, Redux, фільтрація, роутинг, живий пошук.

ABSTRACT

The purpose of the work - analysis of existing monitoring systems, development of a new system for monitoring the priorities of KPI entrants, which will show in real time detailed information on the admission of a particular student sorted by priorities, development of a web application that will have a user-friendly interface and demonstrate work systems.

To solve this problem, a list of tasks was formed, which determined the structure of the study:

- Analyze the subject area, identify tools for software implementation
- Create a user-friendly and minimalist UI
- Creation of a wet database for a monitoring example in JSON format
- Development of logic for sorting by priority and live search
- Development of routing for search with page reloading
- Testing the program and finding bugs
- Eliminate bugs and improve specific parts of the table output interface found during testing

HTML / CSS and Javascript programming languages were used to solve the tasks, and algorithms were used to sort students by priority. Asynchronous queries to the server using the Redux-Saga library and simulating the database itself using the Redux library in JSON format were added to the power database. The system was written on the React framework. System development is carried out in the WebStorm 2020 environment.

The note contains 79 pages, 25 figures, 2 appendices and 15 links.

Keywords: introductory company, monitoring system, MVC, React, Redux, filtering, routing, live search.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ

Термін, скорочення	Значення
БД	База даних
КПІ	Київський політехнічний інститут
ВНЗ	Вищий навчальний заклад
ES6	ECMAScript 6
ЗНО	Зовнішнє незалежне оцінювання
JS	JavaScript
MVC	Modal View Controller
DOM	Document Object Model
URL	Uniform Resource Locator

ВСТУП

З кожним днем програмне забезпечення впливає на наше життя все більше і більше. Виникають нові проблеми, які слід вирішувати шляхом автоматизації.

Сфера навчання в інтернеті набуває більшої популярності. Оскільки основною складовою для успіху будь-якого проекту є, звичайно задоволеність користувачів, інженери щодня працюють над новими системами, рішеннями, що можуть покращити їх продукт і зробити більш простий та мінімалістичний інтерфейс для споживача. В результаті цього ми вирішили автоматизувати деякі процеси для абітурієнтів та створити систему, яка б дозволяла в режимі реального часу здійснювати моніторинг стану пріоритетів для студентів Київського Політехнічного Інституту ім. Ігоря Сікорського, яка поліпшує життя для молодих вступників до нашого вузу та передбачає виведення інформації у вигляді таблиць із багатьма факторами для моніторингу себе на рівні з іншими студентами при вступі, адже конкуренція – це двигун прогресу, а саме в нашому проекті можна отримати інформацію щодо своїх конкурентів при вступі і оцінити свої шанси.

Кількість абітурієнтів до КПІ також тільки зростає, оскільки університет з року в рік випускає тисячі затребуваних спеціалістів в сфері ІТ. З такими результатами кількість охочих вступити до університету стрімко зростає. Поріг вступу на державне навчання, також, зростає разом із цим. Для автоматизації процесів вступу абітурієнтів до КПІ та інших ВНЗ було створено безліч інформаційних систем, щоб покращити електронний рівень доступу, що дуже спрощує життя для молодих спеціалістів при вступі до КПІ.

Система моніторингу - це підклас системи фільтрації інформації, яка прагне передбачити “рейтинг” або “уподобання”, який користувач надав би предмету.

Метою даної роботи є створення системи моніторингу, на основі аналізу

пріоритетів абітурієнтів, та виведення інформації про кожного в новий мінімалістичний інтерфейс.

Користувач може розглядати всі свої пріоритети до вступу, а також пріоритети інших абітурієнтів. Аналізуючи ці дані, можна передбачити багато факторів.

Було проведено дослідження та зроблено висновок про складність та актуальність розробки програмного продукту для надання персоналізованих рекомендацій в реальному часі.

1 ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ СИСТЕМИ МОНІТОРИНГУ СТАНУ ПРІОРИТЕТІВ ВСТУПНИКІВ ДО КПІ ІМ. ІГОРЯ СІКОРСЬКОГО

Використовуючи теоретичний матеріал, наданий керівником дипломної роботи, реалізувати систему моніторингу стану пріоритетів вступників до Київського інституту ім. Ігоря Сікорського, за допомогою якої можна проаналізувати доцільність використання такої системи у власному інституті для зручності абітурієнтів та їх батьків. Система повинна аналізувати кожного потенційного студента та пріоритет, який вказав студент на певний факультет чи спеціальність і відображати повну відфільтровану інформацію щодо цього студента та всіх його пріоритетів на вступ у реальному часі.

В результаті розробки програмного продукту, користувачу буде надана можливість опрацьовувати дані різних студентів, що подали заяви на вступ саме до КПІ. Програму можна використовувати для дослідження інформації стосовно свого вступу, так і для дослідження вступу інших осіб з детальною інформацією відфільтровану відносно пріоритетів.

Потенційний користувач може задавати параметри для пошуку в системі, щоб дослідити процес вступу пошукової особи. Для нього виводяться таблиця з повною інформацією про кожну особу Користувач може також отримати додаткові функції, які розміщені за посиланнями на сайті. І з урахуванням моніторингу відносно інших студентів орієнтуватися на потрібні йому пріоритети, або ж просто досліджувати власну інформацію по вступу до КПІ і отримувати найбільш оновлену інформацію онлайн. А за рахунок цього користувач зможе проаналізувати доцільність вступу до того чи іншого факультету/спеціальності.

Розроблена система повинна забезпечувати наступні можливості для перегляду інформації в таблиці щодо:

- ПІБ абітурієнта
- Пріоритет заяви
- Місця
- Конкурсний бал
- Середній бал документа про освіту
- Складові конкурсного балу, коефіцієнти
- Факультет
- Спеціальність(напрям) / Спеціалізація
- Вступ за квотою
- Статус оригіналів документів

Для розробки продукту була використана мова програмування Javascript. Розробка графічного інтерфейсу користувача відбувалась на основі React JS, Redux, Javascript, HTML5 та CSS.

Метою розробки системи моніторингу є створення програмного продукту, що дасть змогу отримати дані, спираючись на які, користувач, маючи кілька параметрів, зможе оцінити раціональність вступу до певного факультету та переглянути свою конкуренцію. Це дасть зважено та чітко проаналізувати всі найменші аспекти вступу. Також за допомогою цієї системи користувач матиме можливість більш компетентно оцінити статус нашого вищого навчального закладу, адже далеко не у всіх закладах є така можливість.

Мета цього проекту полягає в тому, щоб дати змогу звичайному користувачу ознайомитись з ресурсами по вступній компанії до ВНЗ. Цей проект дає змогу користувачам підібрати найліпший для них варіант з урахуванням всіх факторів, які необхідні при вступі. А також об’єктивно та наглядно порівняти запропоновану інформацію щодо інших студентів.

2 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ ТА АНАЛІЗ ПРОБЛЕМИ

Система моніторингу стану пріоритетів вступників до Київського Політехнічного Інституту ім. Ігоря Сікорського — це технічна система, що забезпечує надання інформації про вступ до ВНЗ в реальному часі.

Наразі існує дуже мало сайтів, що пропонують аналогічну концепцію для забезпечення інформації щодо вступу, але, нажаль, не кожен університет має змогу дозволити собі таку систему для забезпечення свого статусу та надання студентам доцільних даних про свій ВНЗ. Кожний з них має функціонал, що забезпечує користувачів загальною інформацією по всій території України, але не суто про окремий ВНЗ, як це робимо ми. Для розрахунку інформації стосовно особи, пропонується ввести ПІБ. Але такі рішення не є точними і можуть запропонувати лише приблизне значення, тому в нашій системі ми додатково додали в пошук інформацію щодо року вступу, щоб оптимізувати пошук для наших користувачів. Також, аналогічні системи не дозволяють обрати студента конкретного ВНЗ. Таких систем дуже мало і вони забезпечують малий функціонал.

.

2.1. Система моніторингу

Основний принцип системи моніторингу полягає в тому, щоб дозволити користувачам систематично збирати дані, обробляти та поширювати інформацію. Система моніторингу дозволяє нам вимірювати тенденції різних показників на основі даних, зібраних у цій галузі. Система моніторингу є життєво важливою для підтримки після ліквідації наслідків катастроф та їх відновлення.

Для чого потрібен моніторинг? Дуже важливо знати сильні та слабкі сторони своєї програми та надати достатню інформацію особам, які приймають рішення та ініціативи щодо покращення якості програми, а також це дозволяє оцінити очікувані цілі та результати. Іншими словами, моніторинг гарантує, що діяльність йде на правильний шлях, перевіряючи їх, вимірюючи прогрес у досягненні цілей, виявляючи проблеми під час їх виникнення, визначаючи сильні сторони, які можна наростити. Моніторинг збирає інформацію про доступ користувача щодо використання та задоволення результатами роботи.

Що потрібно для моніторингу? Ефективна система моніторингу повинна мати такі компоненти:

- Базова інформація
- Вибір показників, пов'язаних з діяльністю, результатами та цілями
- Інструменти для збору інформації
- Збір інформації
- Обробляти інформацію
- Аналіз інформації
- Представлення та повідомлення результатів відповідними способами
- Використання інформації

Моніторинг відбуватиметься на трьох рівнях:

- рівень громади
- Районний рівень
- Національний рівень

Огляд та оцінка:

Основна мета огляду - більш детально ознайомитись з проектом, ніж це можливо шляхом регулярного моніторингу. Регулярні офіційні та поглиблені

середньострокові огляди здійснюватимуться зовнішніми консультантами за підтримки співробітників партнерства для розгляду будь-якого аспекту проекту. Мета оцінки полягає в тому, щоб переглянути, ефективність програми, а також зміни або вплив, які спричинили в житті цільових користувачів та громади, і як ці зміни відбуватимуться протягом більш тривалих періодів. Під час оцінки будуть встановлені основні показники ефективності та заходи. Іншими словами, оцінка, спрямована на те, щоб знати, чи досягнута встановлена мета. Оцінювання також буде проведено зовнішніми консультантами в кінці Програми, що буде дуже формальним та структурованим вправою, ніж огляд.

Система моніторингу безпосередньо пов'язана з управлінням за цілями та моніторингом ключових показників ефективності. Це також може допомогти в обробці конкретної інформації для прийняття рішень.

3. ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Важливим чинником, під час розробки програмного продукту, є вибір засобів програмної реалізації та технологій. Середовищем розробки інформаційної системи для моніторингу стану пріоритетів вступників до Київського Політехнічного Інституту ім. Ігоря Сікорського було обрано WebStorm. Для створення графічного інтерфейсу системи використовувався веб-інтерфейс, з використанням JavaScript, CSS, HTML5, SCSS, SASS, React JS та Redux.

Для розробки алгоритмів використовувалась мова програмування JavaScript.

Для зчитування вхідної інформації в систему моніторингу пріоритетів вступників була використана бібліотека Axios, щоб робити асинхронні запити до моделі JSON. А для побудови таблиць та їх відображення у системі моніторингу – Material UI.

3.1. Вибір архітектури програмного комплексу

Для реалізації поставленої задачі була розроблена комплексна архітектура, яка складається з декількох компонентів: мокова бд в форматі JSON з імітацією моделі проекту, за допомогою бібліотеки Redux, та клієнт для взаємодії з всіма компонентами написаний на фрайморці React.

Головним у програмному комплексі є клієнт(фронт-енд частина), він виступає в ролі “моста” між іншими компонентами системи, об’єднувавши їх. Він також містить у собі основну бізнес логіку та логіку асинхронного запросу до JSON файлу з данними, на ньому відбувається вся взаємодія з користувачем. Клієнтський компонент реагує на дії користувача. Він отримує вказівки про подальші дії і виводить результат отриманий з Redux-store(фронт-енд засіб для імітації бд).

Вся система написана на новітніх технологіях React-Redux, що дозволяє без проблем імітувати базу даних на стороні клієнту. Завдяки цим технологіям був реалізований також живий пошук та сортування при виводі даних в таблиці, імітація localStorage завдяки бібліотеці React-Router, для збереження даних після перезавантаження сторінки та бібліотека MaterialUI для мінімалістичного та зрачного інтерфейсу для таблиць.

3.2. Опис архітектури клієнтського застосунку

Для реалізації клієнтського компоненту системи, було вибрано фреймворк, в основу якого було покладено шаблон проектування MVC (Рисунок 3.2.1).

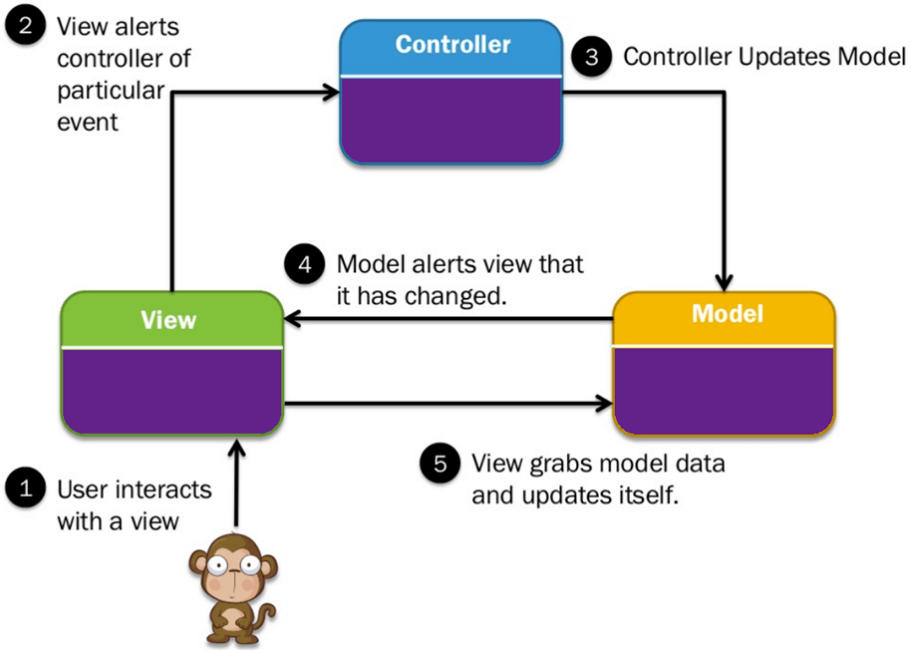

Рисунок 3.2.1 – Візуалізація роботи MVC

Model-View-Controller (MVC) це архітектурна схема, яка розділяє додаток на три основні логічні компоненти Model, View і Controller. Звідси абревіатура MVC. Кожен компонент архітектури побудований для обробки конкретного аспекту розвитку програми. MVC відокремлює бізнес-логіку та рівень

презентації один від одного. Традиційно його використовували для графічних інтерфейсів користувача (GUI) настільних ПК. У наш час архітектура MVC стала популярною для розробки веб-програм, а також мобільних додатків.

Історія MVC:

- Архітектура MVC вперше обговорювалася у 1979 році.
- Модель MVC вперше була представлена в 1987 році мовою програмування Smalltalk.
- MVC вперше був прийнятий як загальна концепція, у статті 1988 року
- Останнім часом модель MVC широко використовується в сучасних веб-додатках.

Особливості MVC:

- Високопробний, розширюваний та підключений каркас
- Забезпечує повний контроль над вашим HTML, а також над вашими URL-адресами
- Використовуйте існуючі функції, що надаються ASP.NET, JSP, Django тощо.
- Чітке розділення логіки: Модель, Вид, Контролер.
- Поділ завдань програми, а саме. бізнес-логіка, логіка Ul та логіка введення даних
- Маршрутизація URL-адрес для SEO URL-адрес.
- Потужне URL-відображення для зрозумілих та пошукових URL-адрес
- Підтримка розробки тестових програм (TDD)

Три важливі MVC компоненти:

- Модель: Вона включає всі дані

- Перегляд: представлення даних користувачеві або обробка взаємодії з користувачем
- Контролер: інтерфейс між компонентами Model і View

View - вигляд це частина програми, яка представляє представлення даних.

Перегляди створюються за даними, зібраними з даних моделі. Перегляд вимагає, щоб модель надавала інформацію так, щоб вона нагадувала вихідну презентацію користувачеві.

Вигляд також представляє дані з чатів, діаграм та таблиці. Наприклад, будь-який перегляд клієнта буде включати всі компоненти інтерфейсу користувача, наприклад текстові поля, спади тощо.

Controller - контролер це та частина програми, яка обробляє взаємодію з користувачем. Контролер інтерпретує введення миші та клавіатури від користувача, інформуючи модель та вигляд, щоб змінити, у відповідних випадках.

Контролер надсилає команди моделі для оновлення свого стану (наприклад, збереження конкретного документа). Контролер також надсилає команди до пов'язаного з ним перегляду для зміни подання перегляду (наприклад, прокрутка певного документа).

Model - компонент моделі зберігає дані та пов'язану з цим логіку. Він представляє дані, що передаються між компонентами контролера або будь-якою іншою пов'язаною бізнес-логікою. Наприклад, об'єкт Controller отримає інформацію про клієнта з бази даних. Він маніпулює даними та надсилає назад до бази даних або використовує їх для надання тих же даних. Він відповідає на запит від View, а також відповідає на вказівки контролера щодо оновлення. Це також найнижчий рівень структури, який відповідає за збереження даних.

- Підсумок:
- MVC це архітектурна схема, яка розділяє додаток на 1) Модель, 2) Вид і 3) Контролер
- Модель: Вона включає всі дані та пов'язану з цим логіку
- Вид: представлення даних користувачеві або обробка взаємодії з користувачем
- Контролер: інтерфейс між компонентами Model і View
- Архітектура MVC вперше обговорювалася у 1979 році
- MVC це дуже перевірена, розширювальний архітектурний паттерн
- Деякі популярні рамки MVC це Rails, Zend Framework, CodeIgniter, Laravel, Fuel PHP тощо.

Для реалізації MVC в моєму React-Redux проекті було організовану таку систему (Рисунок 3.2.2).

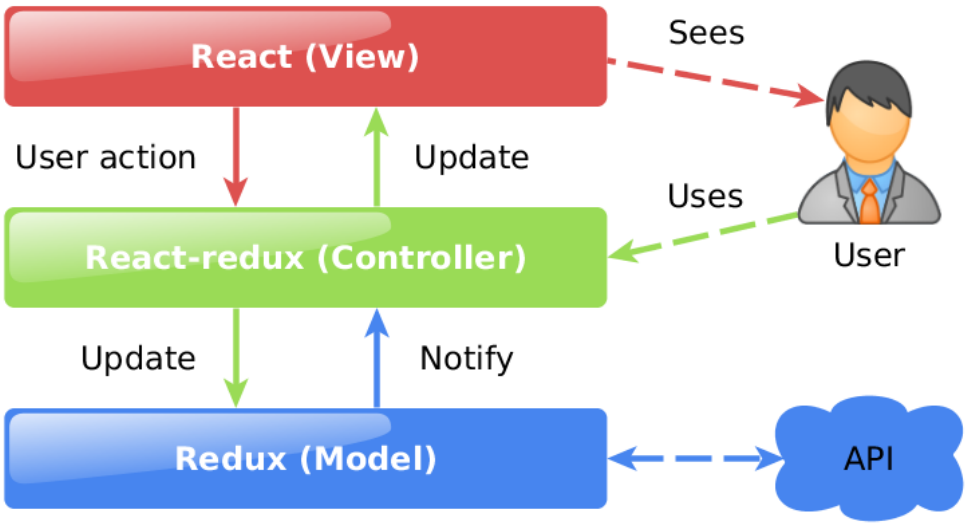

Рисунок 3.2.2 – Візуалізація роботи MVC в React-Redux проекті

Веб-каркаси для мого проекту: Redux, JSON, React JS, JavaScript, React-Redux, Redux-Saga, React-Router, MaterialUI, HTML/CSS.

3.3. Середовище розробки WebStorm

WebStorm — комерційне інтегроване середовище розробки для різних мов програмування (JavaScript, HTML5, CSS3, CoffeeScript, Dart, TypeScript, LESS, Sass, SCSS, Stylus та ін.) від компанії JetBrains. Система поставляється у вигляді розширеної по функціональності платної версії. Є аналогом безкоштовної IDE «VS Code».

Основні можливості:

- Модифікація файлів .css, .html, .js з одночасним переглядом результатів (в деяких джерелах ця функціональність називається «редагування файлів на льоту» або «в реальному часі» або «без перезавантаження сторінки»)
- підтримка HTML5
- Підтримка JSDoc
- підтримка Node.js
- Можливості Zen Coding і Emmet
- Налагодження коду на JavaScript
- Віддалене розгортання по протоколах FTP, SFTP, на монтованих мережевих дисках з можливістю автоматичної синхронізації
- Інтеграція з системами управління версіями: Subversion, Git, GitHub, Perforce, Mercurial, CVS підтримуються з коробки з можливістю створення списків змін і відкладених змін
- Інтеграція з системами стеження за вадами

Я використовую JetBrains WebStorm, що є IDE (інтегрованим середовищем розробки) для розробки JavaScript. Крім того, я отримав від своєї компанії ліцензію WebStorm, і тому можу ним користуватися без будь-яких обмежень.

- Перевірка коду - WebStorm забезпечує надійний, швидкий та гнучкий аналіз статичного коду. Цей аналіз виявляє помилки мови та часу виконання, пропонує виправлення та вдосконалення. Він також індексує весь ваш проект і може, наприклад, виявити всі невикористані методи, змінні та інше. Ви також можете виявити невикористані методи в методах JavaScript, використовуючи ESLint з правилами no-unused-vars. Це може мати величезний вплив на якість коду великого коду.
- WebStorm забезпечує всю функціональність для складної роботи з git поза коробкою. Ви можете створювати файли, переглядати зміни та вирішувати конфлікти за допомогою інструмента візуального розрізнення та злиття прямо в IDE.
- WebStorm автоматично відстежує всі зміни, які ви внесли у свої файли, і тому захищає вас від випадкових втрат цих змін. Ви можете ознайомитись з історією файлів і каталогів і зробити зворотний зв'язок. Це може бути корисно, якщо ви, наприклад, випадково натиснули git і перезаписали файли навіть на віддаленій гілці.
- Використовуючи WebStorm, у вас вже є все доступне за замовчуванням, і, наприклад, для React JS потрібно просто натиснути «Налагодження програми», і ви можете встановити точки розриву в редакторі і переглянути змінні.
- Рефакторити свій код набагато краще, використовуючи WebStorm. Ви можете перейменувати компонент, і він оновлює всі назви файлів і звичаїв як у HTML, так і у файлах JavaScript. Загалом, всі IDE JetBrains добре відомі своїми особливостями рефакторингу.
- WebStorm розроблений на Java, і він відчуває себе загалом повільніше, ніж код VS. Я б не сказав, що це критично повільніше, але різниця швидкостей помітна.

- Він надає перевірки, які базуються на рекомендаціях з інструкцій щодо доступності веб-контенту (WCAG), які допомагають написати більш доступний HTML-код.

Вам потрібно заплатити за ліцензію WebStorm, якщо ви не обрали одну з безкоштовних ліцензій, доступних для проектів з відкритим кодом, студентів, викладачів, допомоги в класі чи навчальних курсах, шкіл кодування та завантажувальних таборів.

Як висновок можна сказати, що WebStorm - універсальний IDE з хорошим базовим функціоналом без необхідності встановлювати багато додаткових плагінів.

Якщо ви надаєте пріоритет швидкості, віддайте перевагу використанню програмного забезпечення з відкритим кодом або просто хочете швидко редагувати деякі файли конфігурації, тоді вам слід скористатися кодом WebStorm.

3.4. Веб-інтерфейс

Веб-інтерфейс - це взаємодія між користувачем та програмним забезпеченням, що працює на веб-сервері. Користувацький інтерфейс - це веб-браузер та веб-сторінка, яку він завантажив та надав.

У контексті веб-сайтів веб-інтерфейс - це сторінка, з якою користувачі взаємодіють, коли сайт повністю завантажується у веб-браузер. Веб-сайт - це сукупність кодів, але цей код не підходить для взаємодії з користувачем. Для того, щоб відвідувачі могли використовувати сайт, код повинен мати веб-інтерфейс, з яким користувачі можуть легко взаємодіяти.

Багато в чому веб-дизайн полягає у створенні веб-інтерфейсу, який

дозволяє відвідувачам командувати веб-сайтом діяти так, як це корисно.

З цієї причини важливо створити привабливий та інтуїтивний веб-інтерфейс. Користувачі повинні мати можливість знайти інформацію, яку вони шукають. Цю нішу часто називають дизайном інтерфейсу користувача (User Interface).

Компоненти веб-інтерфейсу

При розробці веб-сайту чи блогу важливо пам’ятати про певні компоненти інтерфейсу веб-сайту. Давайте розглянемо деякі ключові елементи, якими користуються провідні дизайнери інтерфейсу для створення ефективних веб-сайтів:

- UX - коли відвідувач приходить на ваш сайт, він повинен знати, що інформація знаходяться в потрібному місці. Це означає, що ваш дизайн інтерфейсу повинен чітко вказувати особу вашого веб-сайту чи блогу. Відвідувачі також повинні мати можливість швидко знайти посилання на відповідні частини вашого веб-сайту.
- Легкий доступ - хоча це тісно пов'язане з вищезазначеним фактором, легкий доступ є важливою складовою для будь-якого хорошого дизайну інтерфейсу. Потрібно переконатися, що відвідувачі Вашого сайту мають інтуїтивний потік кліків, щоб знайти те, що вони шукають.
- Зробіть максимально легким доступ до найважливіших речей. Тільки тому, що ви знаєте, де все, не означає, що ваші відвідувачі будуть.
- UI - Зрештою, ви б не довіряли сайту, який виглядає непрофесійно. Створення привабливого, професійного дизайну може допомогти вселити довіру до вашого веб-сайту та бренда. Непрацювання дизайну веб-сайтів може коштувати клієнтам, перш ніж вони навіть переглянуть ваші товари чи вміст.
- Уникайте перестаратися - якщо ваш дизайн поганий або має багато лишніх функцій, це може вплинути на ефективність вашої сторінки.

- Швидкість сторінки - важлива складова SEO. Якщо ви хочете отримати високий рейтинг у результатах пошуку Google, важливо, щоб ваш сайт працював ефективно. Користувачі також вважають за краще взаємодіяти з веб-сайтами, які швидко завантажуються, тому уповільнення швидкості сторінки збільшить показник відмов.

3.5. Бібліотеки React JS & Redux

React JS - це бібліотека JavaScript, яка використовується в веб-розробці для створення інтерактивних елементів на веб-сайтах.

JavaScript (або JS) - це сценарна мова, яка використовується для створення та керування динамічним веб-контентом.

Динамічний веб-вміст включає такі речі, як анімована графіка, слайдшоу для фотографій та інтерактивні форми.

Щоразу, коли ви відвідуєте веб-сайт, де на екрані переміщуються, оновлюються чи іншим чином змінюються зміни, не вимагаючи перезавантажувати веб-сторінку вручну, це JS.

JavaScript є надзвичайно важливою мовою кодування, яка використовується для додавання анімованих та інтерактивних функцій на веб-сайти чи веб-додатки (поверх основних, статичних структур, утворених такими мовами, як HTML та CSS).

React - бібліотека JavaScript, яка спеціалізується на допомозі розробникам у створенні інтерфейсів користувача. Що стосується веб-сайтів та веб-додатків, то користувальницькі інтерфейси - це набір екранних меню, панелей пошуку, кнопок і всього іншого, з ким користується веб-сайт чи додаток.

React використовується виключно для програмування на стороні клієнта (побудова речей, які користувач побачить на екрані у вікні свого браузера), завдяки чому React JS є бібліотекою прикордонних клієнтів.

До React JS, розробники застрягли створювати інтерфейси вручну за

допомогою «ванільного JavaScript» (розробник розмовляє самостійно для сирої мови JavaScript) або з менш орієнтованими на інтерфейс попередниками React, як jQuery. Це означало довші часи розробки та багато можливостей для помилок. Так, у 2011 році інженер Facebook Джордан Уолк створив React JS спеціально для покращення розвитку інтерфейсу.

Крім надання коду бібліотеки React для багаторазового використання (економія часу на розробку та зменшення шансів на помилки кодування), React має дві основні функції, які додають до її звернення для розробників JavaScript:

- JSX - Основою будь-якого базового веб-сайту є HTML-документи. Веб-браузери читають ці документи та відображають їх на комп'ютері, планшеті чи телефоні у вигляді веб-сторінок. Під час цього процесу браузери створюють щось, що називається Document Object Model (DOM), дерево уявлення про те, як влаштовано веб-сторінку. Потім розробники можуть додавати динамічний контент у свої проекти, змінюючи DOM на таких мовах, як JavaScript. JSX (скорочення JavaScript eXtension) - розширення React, яке дозволяє веб-розробникам змінювати свій DOM за допомогою простого коду в стилі HTML. І оскільки підтримка браузера React JS поширюється на всі сучасні веб-браузери, JSX сумісний з будь-якою платформою браузера, з якою ви могли б працювати. Це не лише питання зручності, але використання JSX для оновлення DOM призводить до значних покращень продуктивності сайту та ефективності розробки.
- Віртуальна DOM - Якщо ви не використовуєте React JS (та JSX), ваш веб-сайт використовує HTML для оновлення DOM (процес, який робить зміни на екрані, що потребують оновлення сторінки вручну). Це чудово працює для простих, статичних веб-сайтів, але для динамічних веб-сайтів, які передбачають важку взаємодію з користувачем, це може стати проблемою (оскільки весь DOM потребує перезавантаження кожного разу, коли користувач натискає

функцію, яка вимагає оновлення сторінки). Однак якщо розробник використовує JSX для маніпулювання та оновлення своєї DOM, React JS створює щось, що називається Virtual DOM. Virtual DOM (як випливає з назви) - це копія DOM сайту, і React JS використовує цю копію, щоб побачити, які частини фактичного DOM потрібно змінити, коли подія відбувається (наприклад, користувач натискає кнопку). Скажімо, користувач вводить коментар у форму публікації блогу і натискає кнопку "Коментар". Без використання React JS, весь DOM повинен був би оновитись, щоб відобразити цю зміну (використовуючи час та потужність обробки, необхідні для цього оновлення). З іншого боку, React сканує Virtual DOM, щоб побачити, що змінилося після дії користувача (у цьому випадку додається коментар) і вибірково оновить цей розділ лише DOM. Таке селективне оновлення вимагає менше обчислювальної потужності та меншої кількості часу на завантаження, що може здатися не дуже схожим на один коментар у блозі, але, коли ви почнете думати про всю динаміку та оновлення - ви зрозумієте це дуже важливо.

Redux - одна з найпопулярніших бібліотек у розробці на сьогоднішній день. Однак багато людей плутаються з приводу того, що це таке і які його переваги. Як зазначено в документації, Redux є передбачуваним контейнером стану для додатків JavaScript. Перефразовуючи це, це архітектура потоку даних додатків, а не традиційна бібліотека або рамки, такі як Underscore.js та AngularJS.

Redux використовується здебільшого для управління додатками. Підсумовуючи це, Redux підтримує стан всієї програми в одному незмінному дереві стану (об'єкта), яке не можна змінити безпосередньо. Коли щось змінюється, створюється новий об’єкт (використовуючи дії та редуктори).

Переваги Redux:

- Передбачуваність результату - завжди є одне джерело істини, без плутанини щодо того, як синхронізувати поточний стан з діями та іншими частинами програми.
- Ремонтопридатність - наявність передбачуваних результатів і сувора структура полегшує підтримку коду.
- Організація - Redux суворіший щодо того, як слід організовувати код, що робить код більш послідовним та легшим для роботи команди.
- Відображення сервера - Це дуже корисно для оптимізації пошукових систем.
- Інструменти для розробників - Розробники можуть відслідковувати все, що відбувається в додатку в режимі реального часу, від дій до змін стану.
- Спільнота та екосистема - це величезний плюс, коли ви навчаєтесь чи використовуєте будь-яку бібліотеку. Наявність спільноти за Redux робить її ще більш привабливою для використання.
- Простота тестування - Перше правило написання тестового коду - це написання невеликих функцій. Код Redux - це в основному функції, які є саме такими: маленькі, чисті та ізольовані.
- Функціональне програмування - Як було сказано, Redux був побудований на основі концепцій функціонального програмування. Розуміння цих понять дуже важливо для розуміння того, як і чому Redux працює так, як це робиться.

Розглянемо основні поняття функціонального програмування:

- Він здатний трактувати функції як першокласні об'єкти.
- Він здатний передавати функції як аргументи.
- Він здатний керувати потоком за допомогою функцій, рекурсій та масивів.

- Він здатний використовувати чисті, рекурсивні функції вищого порядку, закриття та анонімні функції.
- Він здатний використовувати допоміжні функції, такі як карта, фільтр та зменшення.
- Він здатний поєднувати функції разом.
- Порядок виконання коду не важливий.

Функціональне програмування дозволяє нам писати більш чистий і модульний код. Записуючи менші та простіші функції, які виокремлюються за обсягом та логікою, ми можемо значно полегшити код для тестування, обслуговування та налагодження. Тепер ці менші функції стають кодом для багаторазового використання, і це дозволяє писати менше коду, а менше коду - це добре. Функції можна скопіювати та вставити будь-де без будь-яких змін. Функції, які є ізольованими за обсягом і виконують лише одне завдання, менше залежатимуть від інших модулів у додатку, і це зменшене з'єднання - ще одна перевага функціонального програмування.

Redux активно використовує чисті функції. Чисті функції повертають нове значення на основі переданих їм аргументів. Вони не змінюють існуючі об'єкти; натомість вони повертають нове. Ці функції не залежать від стану, з якого вони викликані, і вони повертають лише один і той же результат за будь-який наданий аргумент. З цієї причини вони дуже передбачувані. Оскільки чисті функції не змінюють жодних значень, вони не впливають ні на область застосування, ні на помітні побічні ефекти, і це означає, що розробник може зосередитись лише на значеннях, які чиста функція повертає.

Більшість розробників асоціює Redux з React.

Однією з причин, чому Redux є дивовижним, є його екосистема. Стільки статей, навчальних посібників, програмного забезпечення та інструментів.

Redux з кожним днем набирає тяги. Його використовують у багатьох компаніях і в багатьох проектах.

3.6. Бібліотека React Router

Користувач повинен побачити, де він знаходиться в поточному моменті часу, а також мати можливість навігації за історією. Сам по собі React маршрутизацію не підтримує, тому для цього й був створенний React-Router.

Компоненти - це серце потужної декларативної моделі програмування React.

React Router - це сукупність навігаційних компонентів, які декларативно складаються разом із вашою програмою. React Router працює там, де відображається React (Рисунок 3.6.1).

```
import React from "react";
import ReactDom from "react-dom";
import { Router, Route, IndexRoute , hashHistory } from "react-router";

import About from "./components/pages/About";
import Main from "./components/pages/Main";
import Other from "./components/pages/Other";
import Layout from "./components/Layout";

const app = document.getElementById("app");

ReactDom.render(
  <Router history = {hashHistory}>
      <Route path ="/" component ={Layout}>
          <IndexRoute component ={ Main }></IndexRoute>
          <Route path = "other" component ={ Other }></Route>
          <Route path = "about" component = { About }></Route>
      </Route>
  </Router>
,app);

```

Рисунок 3.6.1 – структура React-Router

3.7. React/Redux dev tools

Однією з особливостей Redux DevTools є вибір дії в історії і перегляд стека, який його викликав. Він спрямований на вирішення проблеми пошуку джерела подій в списку подій (Рисунок 3.7.1).

Існує також traceLimit параметр, який 10 за замовчуванням використовується для запобігання надмірного використання пам'яті і сериализации великих стеків. Це запобігає використанню невеликої кількості пам'яті і сериализацию великих стеків, а також дозволяє отримувати стеки більшого розміру, ніж обмежена браузером (це перевищить обмеження за замовчуванням, 10 накладене Chrome в Error.stackTraceLimit).

Він може відкрити файл (і перейти до рядків) прямо в вашому редакторі. Досить корисно для налагодження, а також в якості альтернативи, коли неможливо використовувати openResource (для Firefox або при використанні розширення з вікна). Ви можете натиснути кнопку «Налаштування» і включити її, додавши також шлях до кореневого каталогу вашого проекту.

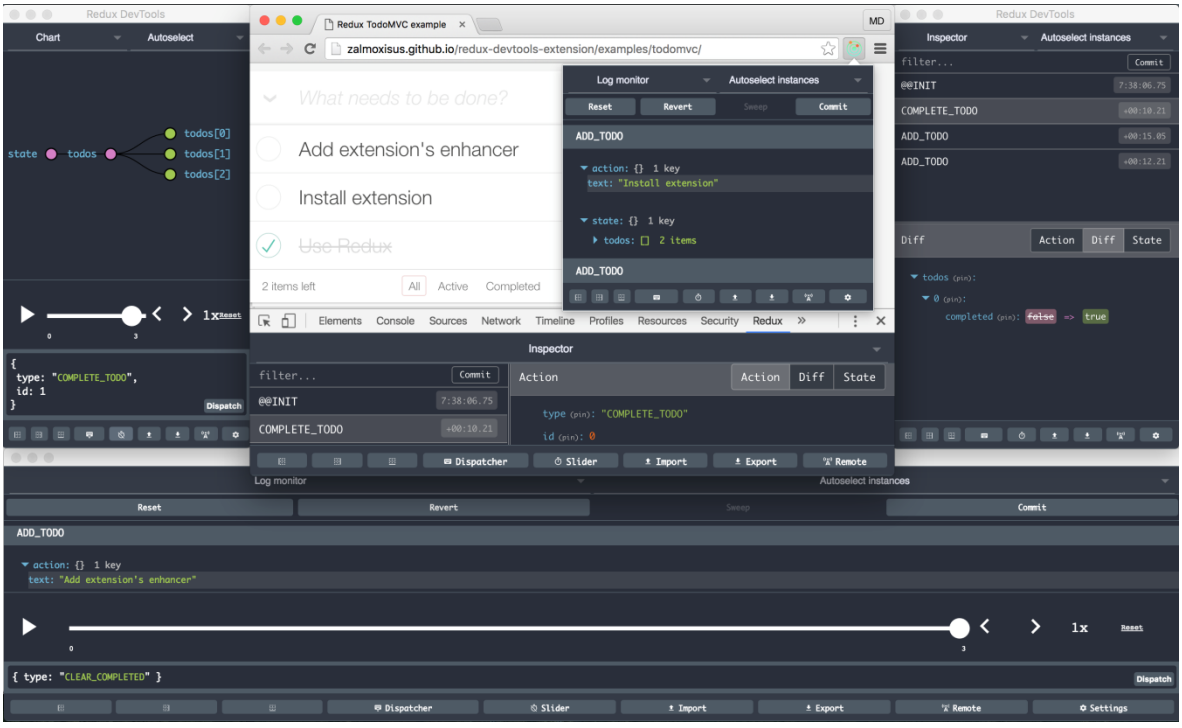

Рисунок 3.7.1 – Redux dev tools

React Developer Tools - це розширення Chrome DevTools для бібліотеки React JavaScript з відкритим вихідним кодом. Це дозволяє вам переглядати ієрархії компонентів React в Chrome Developer Tools.

У Chrome DevTools ви отримаєте дві нові вкладки: «Компоненти» і «профілювальник».

На вкладці «Компоненти» відображаються кореневі компоненти React, які були відображені на сторінці, а також подкомпоненти, які вони в підсумку відобразили.

Вибравши один з компонентів в дереві, ви можете перевірити і відредагувати його поточний реквізит і стан на панелі праворуч.

Якщо ви переглядаєте елемент React на сторінці за допомогою звичайної вкладки Elements, а потім перемикаєтеся на вкладку React, цей елемент буде автоматично вибрано в дереві React.

Вкладка «профілювальник» дозволяє записувати інформацію про продуктивність.

React DevTools доступний як вбудоване розширення для браузерів Chrome і Firefox. Цей пакет дозволяє налагоджувати додаток React в іншому місці (наприклад, в мобільному браузері, у вбудованому веб-поданні, Safari, всередині iframe).

Працює як з React DOM, так і з React Native.

При налагодженні JavaScript в Chrome ви можете перевірити реквізити і стан компонентів React в консолі браузера.

Автономна оболонка також може бути корисна з React DOM (наприклад, для налагодження додатків в Safari або всередині iframe).

Він також пропонує повну підтримку React Hooks, включаючи перевірку вкладених об'єктів.

ReactDevTools надає спосіб фільтрації компонентів з дерева, щоб спростити навігацію по глибоко вкладених ієрархій (Рисунок 3.7.2).

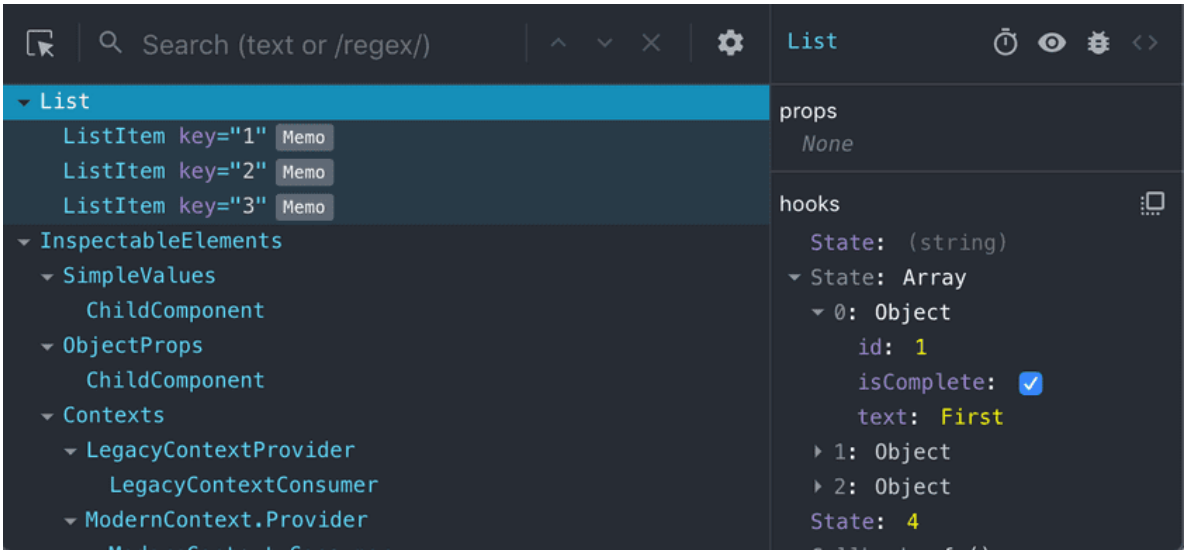

Рисунок 3.7.2 – React dev tools

3.8. Бібліотека MaterialUI

MaterialUI - React компоненти для побудови та легкої веб-розробки. Создайте свій власний дизайн або начнете з Material Design (Рисунок 3.8.1).

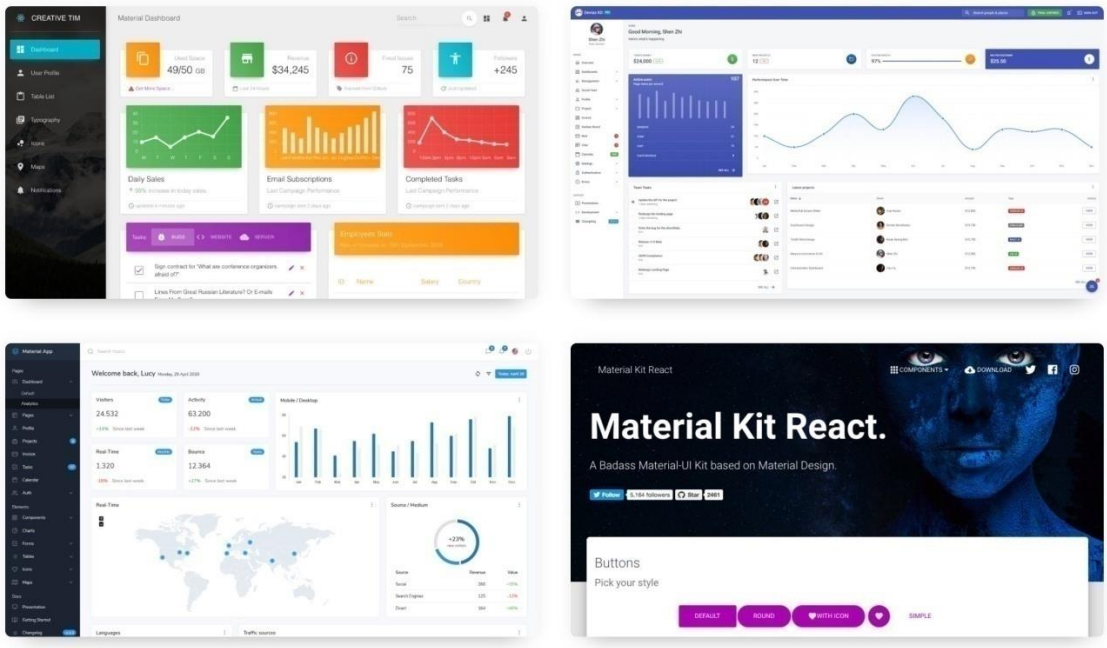

Рисунок 3.8.1 – компоненти MaterialUI

Компоненти MaterialUI працюють без додаткових налаштувань і не забруднюють глобальну сферу застосування

Виявлення представлено в компонентах React для більшої побудови та простого веб-розробки.

MaterialUI буде не тільки реалізовувати рукодіалогічних принципів проектування матеріалів, але і бібліотеку користувальницького інтерфейсу загальнодоступних користувачів. Це використання не підбирає ніякої іншої методології проектування. Це також зазначає, що у нас буде компонентів або комбінацій, які просто не розглядаються в рекомендаціях за проектуванням.

Ми підтримуємо реалізацію всіх низькорівневих інструментів, необхідних для створення багатофункціонального користувальницького інтерфейсу з допомогою React. Після того, як ми вдосконалили специфікацію дизайну матеріалів (яка є досить високою програмою), ви можете використовувати її для свого бізнесу з будь-яким необхідним настроєм стилю. Ми хотіли, щоб компанії перешкоджали використанню матеріалів MaterialUI таким чином, щоб вони відповідали їхньому бренду, зближаючись з філософією матеріалів. Ми не хотіли, щоб вони почували, що їх інтерфейс просто схожий на інший продукт Google.

3.9. Формат JSON

В моєму проекті, найбільш компетентною відносно складності задачі і взаємодії з середовищем інтерфейсу я вирішив обрати за модель бібліотеку Redux, яка імітує стан всього проекту на стороні клієнту. Запити на сервер робляться за допомогою асинхронного Redux-Saga, які в свою чергу зберігають дані в форматі JSON. Використання саме формату json зручно тим, що інформація в даному форматі - це JavasSript масиви і об'єкти, до яких легко можна отримати доступ з клієнтської частини web додатків.

Будь-яка база даних включає в себе набір функцій для запису, читання, оновлення та видалення даних з таблиць. В даному випадку це буде клас з набором методів для управління базою.

Вибірка даних з таблиць відбувається досить швидко. Використання такого методу зберігання інформації добре підходить для зберігання різних налаштувань додатків і тимчасових даних. Як було сказано вище, в цьому так само є великий плюс через те, що до будь-якої інформації, що зберігається в даному виді, можна отримати доступ з клієнтської частини програми.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

В даному розділі міститься обґрунтування архітектурного рішення при розробці системи для моніторингу стану пріоритетів вступників до Київського Політехнічного Інституту ім. Ігоря Сікорського. Також розділ обгрунтовує використання моделі JSON з бібліофтекою Redux та Redux-Saga для асинхронних запросів до сервера з потоковою оптимізацією, функціональність системи, проблеми, які були вирішенні в ході тестування, алгоритми та діаграми програмної реалізації.

Система моніторингу складається з декількох модулів, деякі з них мають функціональні підблоки. Головним модулем є таблиця відображення студентів відсортованих за пріоритетом, які користувач буде бачити після того як вірно зроби запит в систему. Щоб вирішити проблему “холодного пуску”, користувач на головній сторінці буде бачити також додаткові посилання для зручності, за якими він зможе знайти для себе додаткову інформацію й на інших аналогічних сайтах та системах. Дані будуть надходити з сервера, у форматі JSON за допомогою асинхронного запросу, та зберігатись до Redux-store, після чого вже відображатись в інтерфейсі на стороної клієнту. В результаті буде отримано таблицю з багатьма факторами конкретного студента, якщо студента немає в бд, споживачеві буде виведено певне смс, щоб скорегував свій пошуковий запит.

На рисунку 4.1 наведена схема структури системи, на якій розташовані всі програмні модулі.

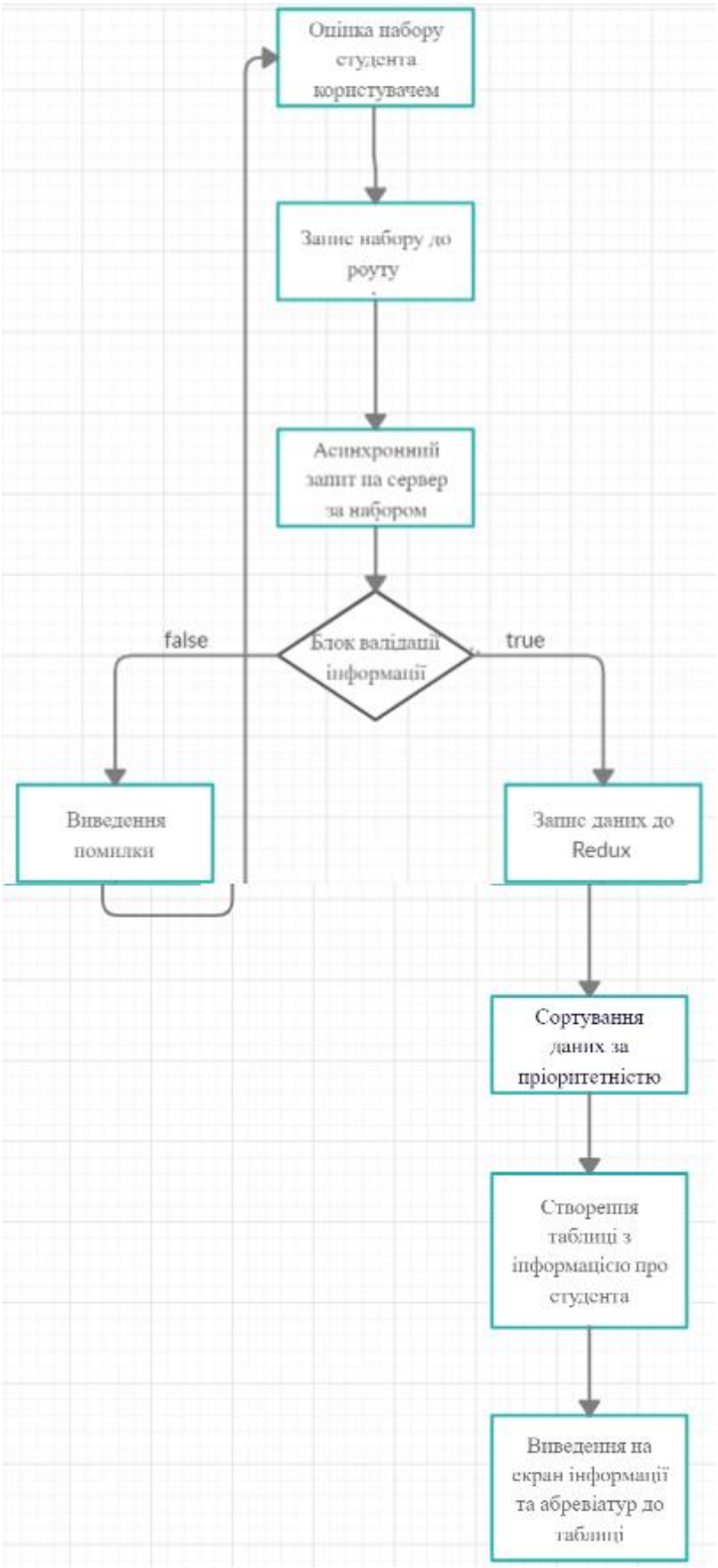

Рисунок 4.1 — Схема структури системи

4.1. Опис функціональності системи

Система моніторингу стану пріоритетів вступників до КПІ ім. Ігоря Сікорського містить у собі головного актора – користувач системи.

На рисунку 4.1.1 представлена діаграма послідовності дій, яка описує функції та дії актора у системі а також клієнту та серверу.

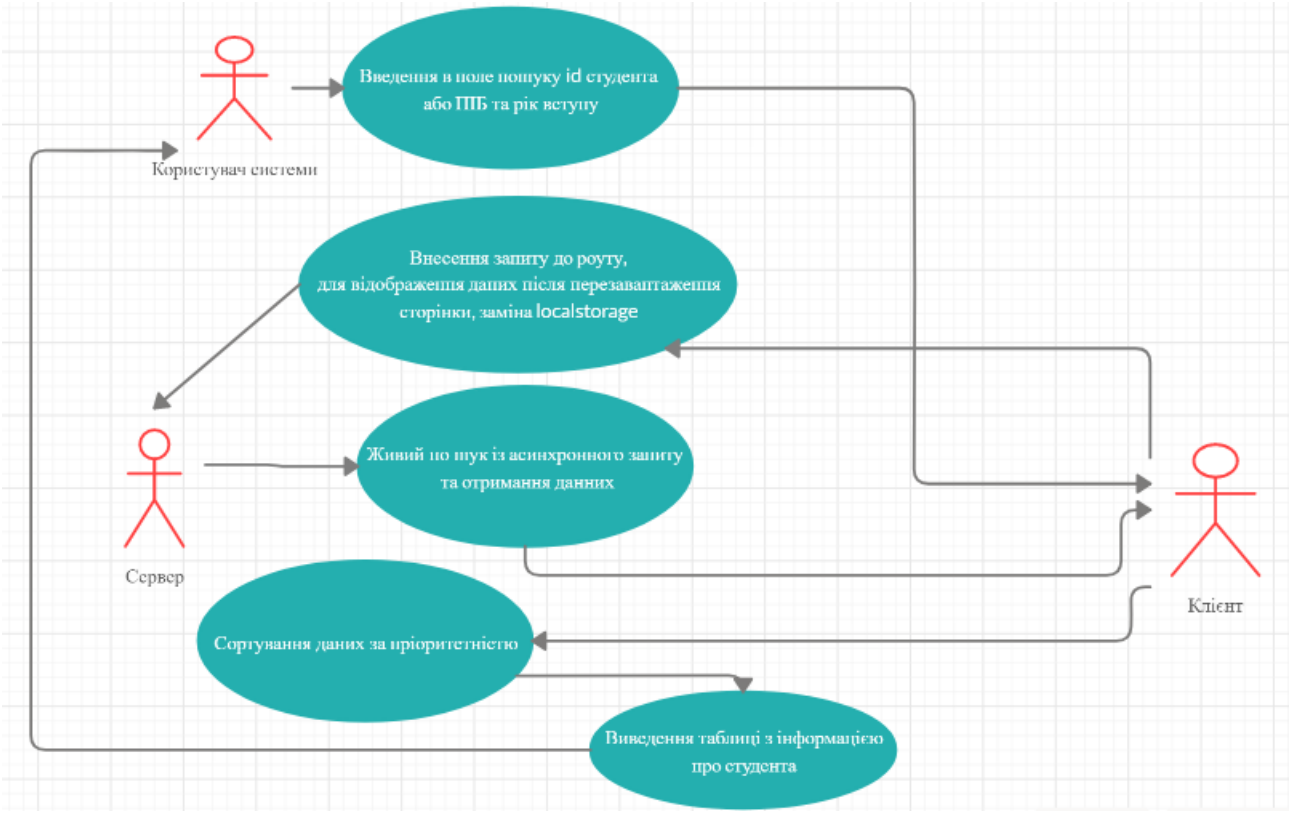

Рисунок 4.1.1 — Діаграма послідовності дій

4.2. Концептуальна модель Redux у форматі JSON з асинхронним запитом на сервер за допомогою Redux-Saga

Було проаналізовано предметну область і визначено, що саме для системи моніторингу буде доцільно використовувати бібліотеку Redux-Saga для оптимізованого та асинхронного запиту на сервер, що саме завдяки асинхронності допомогло налаштувати в інтерфейсі «живий» пошук. Також бібліотекою для збереження отриманих даних було визначено використовувати новітню фронт-енд технологію – Redux, що чудового працює з фрайморком React.

Завдяки саме такому вибору стеку технологій для проекту, він став більш оптимізований і написаний повністю на функціональних компонентах та з новітніми правилами і синтаксисом ES6, що є складовою мови програмування JavaScript.

«Асинхронний запит» - це коли синхронно з ним може виконуватися інший.

Ядро Redux це контейнер стану (state container), який підтримує тільки синхронні потоки даних.

На кожну дію, в сховище (store) надсилається об'єкт, що описує що сталося, потім викликається редюсер (reducer) і стан (state) відразу оновлюється.

Redux використовує проміжні шари (middlewares). Проміжний шар - це шматок коду, який виконується після відправки дії, але перед викликом редюсера.

Проміжні шари можуть з'єднуватися в ланцюжок викликів для різної обробки дії (action), але на виході обов'язково повинен бути простий об'єкт (дія).

Redux-saga це бібліотека націлена робити сайд-ефекти простіше і краще шляхом роботи з сагами.

Саги це дизайн патерн, який прийшов зі світу розподілених транзакцій, де сага управляє процесами, які необхідно виконувати транзакційних способом, зберігаючи стан виконання і компенсуючи невдалі процеси.

В контексті Redux, сага реалізована як проміжний шар (ми не можемо використовувати редюсери бо вони повинні бути чистими функціями), який координує і спонукає асинхронні дії (сайд-ефекти).

Redux-saga робить це за допомогою ES6 генераторів.

Генератори (Generators) - це функції які можуть бути зупинені і продовжені, замість виконання всіх виразів в один прохід.

Коли ви викликаєте функцію-генератор, вона повертає об'єкт-ітератор. І з кожним викликом методу ітератора next() тіло функції-генератора буде виконуватися до наступного yield вираження і потім зупинятися.

Це робить асинхронний код простіше для написання і розуміння.

Redux-saga дозволяє нам отримати об'єкт, який декларує наш намір провести операцію, замість результату виконання самої операції.

Декларативне програмування - це стиль програмування, який намагається мінімізувати або усунути сайд-ефекти, описом що програма повинна робити, замість опису як вона повинна це робити.

На рисунку 4.2.1 - ми бачимо асинхронний запит до серверу.

В 40 рядку ми бачимо перемінну “URL” – в якій ми можемо підставити любий API серверу, що зберігає дані у форматі JSON.

Ключове слово yield викликає зупинку функції-генератора і повертає поточне значення виразу, вказаного після ключового слова yield. Його можна розглядати як аналог ключового слова return в функції-генераторі.

```
// REDUX-SAGA
export function* getStudents() {
  const url = "data.json";
  try {
    const data = yield call(() => {
      return fetch(url).then((res) => res.json());
    });
    yield put({ type: GET_STUDENTS, data });
  } catch (error) {
    console.error(error);
  }
}

export function* parsing() {
  yield takeLatest(GET_STUDENTS_API, getStudents);
}
```

Рисунок 4.2.1 - Асинхронний запит Redux-Saga

Замість виклику асинхронного реквеста безпосередньо, метод call поверне тільки об'єкт, що описує цю операцію і redux-saga зможе подбати про виклик і повернення результатів в функцію-генератор.

Замість відправлення дій (dispatch action) всередині функції-генератора, put повертає об'єкт з інструкціями для проміжного шару (middleware) - відправити дію.

takeEvery – добавить певну дію в список аргументів (тобто дія буде останнім наданим аргументом saga).

takeEvery дозволяє обробляти паралельні дії.

Метод fetch - надає інтерфейс JavaScript для роботи із запитами та відповідями HTTP. Він також надає глобальний метод fetch (), який дозволяє легко і логічно отримувати ресурси по мережі асинхронно.

Після того, як асинхронний запрос на сервер був успішного зробляний, дані з генератора ми записуємо в Redux-store. Store містить всі дерева станів вашої програми. Єдиний спосіб змінити стан всередині нього - це направити (dispatch) action на нього.

Store - це не клас. Це просто об'єкт з деякими методами в ньому. Щоб його створити, передайте вашу кореневу функцію редюсера в createStore.

Виходить, що після нашого запиту на сервер, всі дані які ми отримуємо – ми записуємо в store, для подальшої роботи з даними і імітації бд без додаткових запросів на сервер. Ми можемо змінювати та оновлювати(мутувати) наші дані безпосередньо на стороні клієнту завдяки Redux і цим самим оптимізувати швидкість роботи сайту.

Всі дані ми зберігаємо у форматі JSON для покращення доступу до полів наших даних, що прийшли з сервера.

На рисунку 4.2.2 - зображено стуктуру JSON файлу, з яким ми працюємо в нашій програмі.

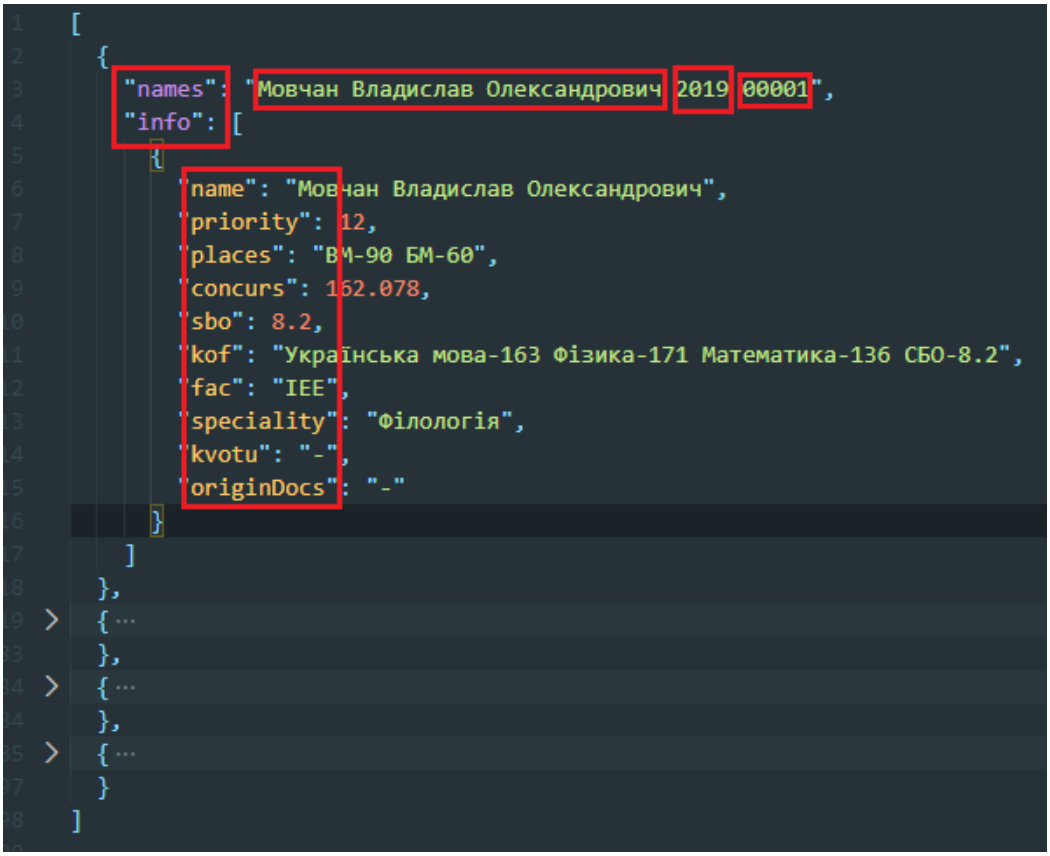

Рисунок 4.2.2 - Стурктура таблиці

Names – це поле для пошуку, воно містить ПІБ студента, якщо ПІБ є однаковий в деяких роках, то вкінці також можна добавити ще рік для більшої точності, але також студента можна знайти й за id, який визначений останнім словом в полі «names».

Info – це поле, в якому зберігається вся інформація про вступ конкретного студента, яка при пошуку відправляєть і виводиться у таблиці з 10 факторами.

Структура самого JSON файлу були використана сааме - массив об’єктів, оскільки це дуже зручно і швидко для нашої системи моніторингу. Ми зможемо додавати таким чином в масив нові об’єкти з такими самими полями(якщо студент має таке саме ПІБ як і інший, який подавав заявку), відрізнити їх зможе тільки рік вступу, але навіть якщо і він однаковий, для цього в нас є id.

Масив містить скільки об’єктів, скільки людей буде зареєстровано при вступі до КПІ. Кожен об’єкт – це новий студент з унікальним id. В кожному такому об’єкті в свою чергу є поле для пошуку цього студента та ще один масив об’єктів, де записується скільки об’єктів, скільки пріоритетів подавав певний студент при вступі та вся інформація щодо вступу по кокретному пріоритету.

Таким чином, при переносі бізнес логіки на клієнт частину, ми спростили загрузку на сайт і зменшили запити на сервер за рахунок Redux, також на стороні клієнту, завдяки якому маємо гнучку архітектуру на фронт-енд частині.

Оновлення наших даних налаштоване таким чином, що при будь-якій взаємодії з сайтом, React-Hook “UseEffect” перевіряє, чи нічого не змінилося з попереднього запросу і зрівнює з нашою копією “students” в Redux-store, якщо якісь зміни є, нові дані записуються в наш Redux-store і наша сторінка перезавантажується уже відносно найновіших даних (Рисунок 4.2.3).

```
useEffect(() => {
  getAPI();
}, [students]);
```

Рисунок 4.2.3 - React-Hook “UseEffect”

React-Hook – це функція, яка запускається при першому рендері компонента і при кожному наступному рендері / оновленні. React спочатку оновлює DOM, потім викликає будь-яку передану функцію useEffect (). В цьому хукі ми можемо поставити залежність, відносно якій і буде відбуватися оновлення сторінки, а завдяки React JS все працює набагто швидше, адже при будь якому оновленні сторінки оновлюється не весь DOM нашого сайту, а React передбачає Virtual DOM, що записує весь DOM в один JavaScript об’єкт і перезаписує тільки найновіші зміни, а оскільки це JS об’єкт – швидкість оновлення набагато швидша і компоненти які не були змінені не оновлюються, що також додає швидкості роботи для сайту.

4.3. Опис таблиці студента

Для доступу до таблиць в програмі використовується бібліотека Material UI, що дає змогу зручно створювати таблиці та отримувати дані з таблиць.

Детальна інформація про їх структуру приведена на рисунку 4.3.1.

ПІБ	П	Місця	∑	СБО	Складові конкурсного балу	Ф	Спец.	Кв	Д
Мовчан Владислав Олександрович	1	ВМ-40 БМ-17 КМ-23	162.078	8.2	Українська мова-163 Фізика-171 Математика-136 СБО-8.2	ТЕФ	Авіоніка	-	+
Мовчан Владислав Олександрович	2	ВМ-12 БМ-6 КМ-6	162.078	8.2	Українська мова-163 Фізика-171 Математика-136 СБО-8.2	ІПСА	Системний аналіз	-	-
Мовчан Владислав Олександрович	3	ВМ-140 БМ-80 КМ-60	162.078	8.2	Українська мова-163 Фізика-171 Математика-136 СБО-8.2	ФІОТ	Програмна інженерія	-	-
Мовчан Владислав Олександрович	4	ВМ-20 БМ-17 КМ-3	162.078	8.2	Українська мова-163 Фізика-171 Математика-136 СБО-8.2	ІХФ	Кібербезпека	-	-
Мовчан Владислав Олександрович	5	ВМ-85 БМ-30 КМ-55	162.078	8.2	Українська мова-163 Фізика-171 Математика-136 СБО-8.2	ПБФ	Журналістика	+	-
Мовчан Владислав Олександрович	6	ВМ-98 БМ-43 КМ-55	162.078	8.2	Українська мова-163 Фізика-171 Математика-136 СБО-8.2	РТФ	Економіка	-	-

Рисунок 4.3.1 - Структура таблиці студента

Із даної таблиці ми бачимо, що всі заявки відсортовані за пріоритетом у зручній по інтерфейсу таблиці.

Таблиці складається з 10 полів, це:

1 ПІБ
2 Пріоритет
3 Місця (всього, бюджет, контракт)
4 Середній бал ЗНО
5 Середній бал документу про освіту
6 Складові конкурсного балу
7 Факультет КПІ
8 Спеціальність
9 Квоти
10 Оригінали документів

4.4. Розробка головної сторінки

Головна сторінка – це основне робоче місце користувача в системі. Вона є елементом компонування всієї системи

Підмодулі головної сторінки:

— модуль пошуку студента;

— модуль додаткових посилань;

— модуль відображення інформації у таблицях;

4.5. Модуль пошуку студента

Даний модуль дозволяє споживачеві здійснювати пошук про конкретного студента.

Цей модуль є «живим» завдяки асинхронному запиту до серверу та пошуку введеного нами слова в БД.

«Живий пошук» - це пошук, який здійснюється миттєво, без перезавантаження сторінки, завдяки новітнім технологіям, які ми використовуємо в проекті. Після введеного нами пошукового слова, нам не потрібного кожного разу перегружати сервер і робити пошук на ньому, відразу при оновленні сторінки виконується асинхронний запит до серверу для отримання інформації про всіх студентів, і коли ми починаємо вводити пошукове слово, пошук йде вже по збереженій в Redux-store копії (Рисунок 4.5.1), що зберігається на стороні клієнту і дозволяє нам оптимізувати роботу сайту і знаходити потрібну нам інформацію в режимі реального часу.

```
const resultsBySearch =
  students.length !== 0
    ? students.filter((data) => {
        const names = data.names.toLowerCase();
        const names2 = decodeURI(
          history.location.search.slice(1).toLowerCase()
        );
        return ~names.indexOf(names2);
      })
    : null;
```

Рисунок 4.5.1 - Алгоритм пошуку по Redux-store

Модуль пошуку був налаштований так, щоб пошук здійснювався не по слову введеному в поле, а щоб слово, введене в пошуковий запит записувалось в посилання нашого сайту і вже по ньому робити пошук в нашому Redux-store.

Також, було використано метод decodeURL(), для забезпечення правильності вводу пошукового слову в модулі пошуку після перезавантаження

сторінки, який замінює кожну послідовність у закодованому URI на відповідні символи.

Це імітує роботу localStorage і також оптимізує роботу сайту, оскільки не потрібно зберігати щось до браузерної БД, щоб після перезавантаження сторінки дані не зникали, тепер після кожного перезавантаження, виконується живий пошук, пошукове слово для якого, вже було записану в наше посилання, ще при першому вводі і результат, який зберігався в Redux-store - буде відразу оновлений на нашому сайті. Ця функція передбаченна завдяки бібліотеці React-Router (Рисунок 4.5.2), що дає нам змогу побачити, де ми знаходиться в поточному моменті часу на нашому сайті, а також є можливість навігації за історією, створювати нові посилання для сайту та користуватися адресною строкою браузера і ще багато нових можливостей.

```
const changeInputHandler = (event) => {
  input = event.target.value.toLowerCase();
  history.push({
    pathname: match.params.category,
    search: input,
  });
};
```

Рисунок 4.5.2 - Реалізація React-Router

Також, перед виведенням на екран інформації про потрібного нам студента, по нашому пошуковому слові ми отримуємо масив об’єктів заявок нашого вступника і по полю “priority” ми здійснюємо сортування від меншого до більшого, після чого вже відправляємо масив для виведення в таблицю для користувача (Рисунок 4.5.3).

```
const priority = resultsBySearch
  ? resultsBySearch.map((el) =>
      el.info.sort((a, b) => (a.priority > b.priority ? 1 : -1))
    )
  : null;
```

Рисунок 4.5.3 - Реалізація сортування по пріоритетності

4.6 Модуль додаткових посилань

Даний модуль (Рисунок 4.6.1) дуже простий і має в своїй складовій звичайні посилання, що дає користувачеві додаткові можливості для зручності. Якщо ж споживач користується нашим сайтом, в головному меню для зручності було розміщенно декілька посилань, за якими він більш всього захоче перейти.

- В першу чергу це сам сайт нашого інституту (https://kpi.ua/)
- Рейтингові списки минулих років (https://abit-poisk.org.ua/rate-review/)
- Спеціальності минулих років (https://abit-poisk.org.ua/specialities2019/)
- Та такаж сама система моніторингу, тільки по всіх ВНЗ (https://abit-poisk.org.ua/#)

```
<a href="https://kpi.ua/" className="logo">
  kpi search
</a>
<nav>
  <ul className="header-navigation">
    <li className="list">
      <a href="https://abit-poisk.org.ua/rate-review/" className="link">
        Рейтингові списки 2019
      </a>
    </li>
    <li className="list">
      <a
        href="https://abit-poisk.org.ua/specialities2019/"
        className="link"
      >
        Спеціальності 2019
      </a>
    </li>
    <li className="list">
      <a href="https://abit-poisk.org.ua/#" className="link">
        Пошук по всіх ВНЗ
      </a>
```

Рисунок 4.6.1 - Реалізація додаткових посилань

4.7 Модуль відображення інформації

Даний модуль дозволяє споживачеві переглядати інформацію про конкретного студента в форматі таблиці (Рисунок 4.7.1).

```
<TableBody>
  {info
    ? info.map((row) => {
        return (
          <TableRow
            hover
            role="checkbox"
            tabIndex={-1}
            key={row.priority}
          >
            {columns.map((column) => {
              const value = row[column.id];
              return (
                <TableCell
                  key={column.id}
                  style={{ width: column.width }}
                >
                  {column.format && typeof value === "number"
                    ? column.format(value)
                    : value}
                </TableCell>
              );
            })}
          </TableRow>
        );
      })
    : null}
</TableBody>
```

Рисунок 4.7.1 - Реалізація модулю відображення інформації

В цьому модулі ми дістаємо дизайн таблиці за допомогою бібліотеки “MaterialUI”, та наші дані з Redux-store, що приходять у вигляді масива об’єктів, дістаємо і виводимо у відповідні рядки.

Колонки нашої таблиці формуються завдяки інтегрованому з MatetialUI компоненту (Рисунок 4.7.2).

```
const columns = [
  { id: "name", label: "ПІБ", width: "20%" },
  { id: "priority", label: "П" },
  { id: "places", label: "Місця", width: "5%" },
  { id: "concurs", label: "Σ" },
  { id: "sbo", label: "СБО" },
  { id: "kof", label: "Складові конкурсного балу", width: "35%" },
  { id: "fac", label: "Ф" },
  { id: "speciality", label: "Спец." },
  { id: "kvotu", label: "Кв" },
  { id: "originDocs", label: "Д" },
];
```

Рисунок 4.7.1 - Реалізація модулю відображення інформації

Кожна колонка має своє унікальне id – що взаємодіє з Redux для отримання даних в колнку з кожної заявки студента. Id та поля в Redux однакові.

Label – це назва колонки при виведенні таблиці в інтерфейс користувача.

Також додатково вказана ширина колонки, там де це потрібно

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З СИСТЕМОЮ

В цьому розділі приведені системні вимоги для роботи з додатком та сценарії роботи користувача з ним.

5.1 Інсталяція та системні вимоги

Для забезпечення коректної та безвідмовної роботи інформаційної системи персональний комп’ютер повинен мати процесор не гірше, ніж Intel ® Pentium ® / Celeron ® / Xeon™ або з тактовою частотою не менше 1,8 GHz або AMD 6 / Turion ™ / Athlon ™ / Duron ™ / Sempron ™ для користувачів процесорів від фірми AMD. Також комп’ютеру користувача повинно бути доступно не менше 2 Gb оперативної пам’яті та графічне ядро не гірше, ніж Intel ® HD Graphics 2000, що еквівалентно графічним картам з об’ємом пам’яті не менше, ніж 128 Mb.

Також можлива взаємодія на телефонних пристроях.

5.2 Інструкція з використання програмного продукту

Головна сторінка системи моніторингу стану пріоритетів вступників до КПІ ім. Ігоря Сікорського має наступний вигляд (Рисунок 5.2.1):

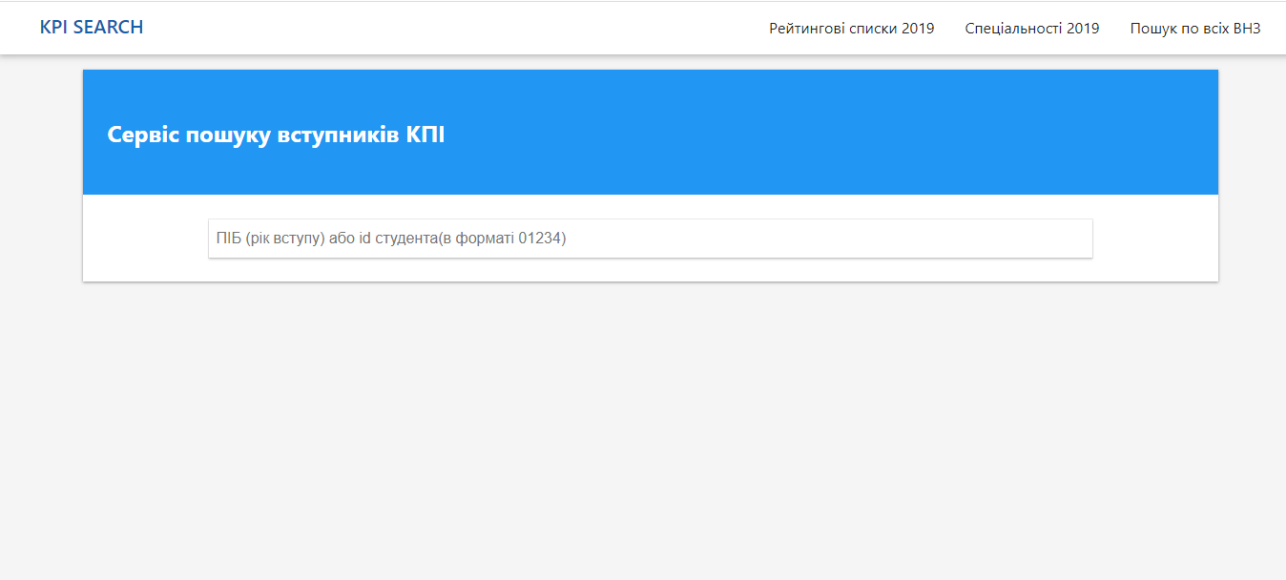

Рисунок 5.2.1 — Головна сторінка

Для початку роботи користувач повинен ввести ПІБ особи, в якої він хоче побачити інформацію про вступ до КПІ і сразу відбудеться «живий» пошук по БД, після чого сайт відобразить користувачу всю відфільтровану по пріоритетам інформацію стосовно пошукової особи пов’язану із вступом до КПІ: (рисунок 5.2.2).

- ПІБ абітурієнта
- Пріоритет заяви
- Місця
- Конкурсний бал
- Середній бал документа про освіту
- Складові конкурсного балу, коефіцієнти
- Факультет
- Спеціальність(напрям) / Спеціалізація
- Вступ за квотою
- Подано оригінали документів

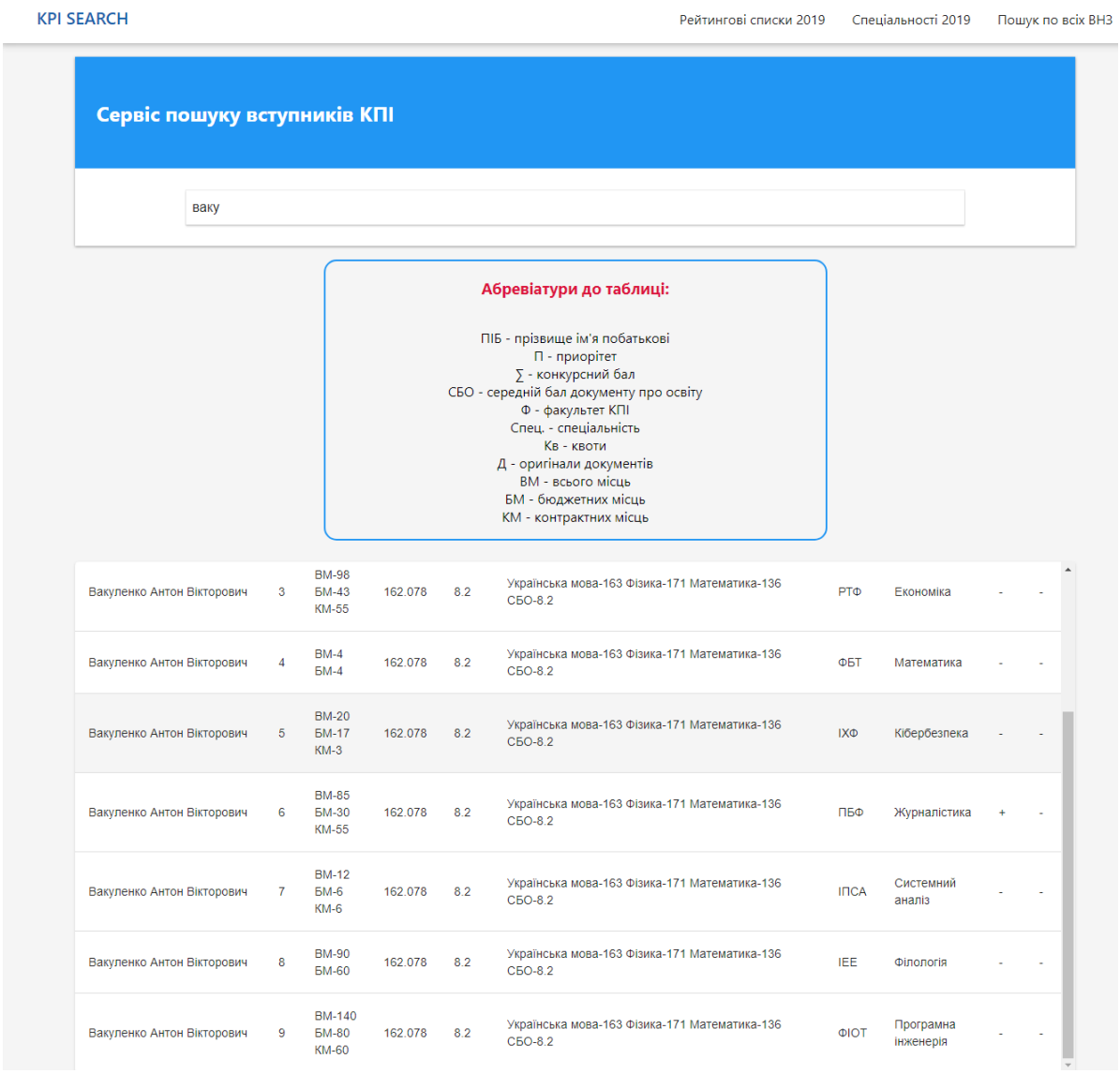

Рисунок 5.2.2 — Відображення інформації при не повному пошуковому слові

На даній сторінці користувач має змогу проаналізувати 10 полів моніторингу процесу вступу студента до КПІ для певного проміжку часу, навіть якщо пошукове слово ще не було введенно повністю, завдяки модулю живого пошуку.

На Рисунку 5.2.3 – ми можемо побачити, який буде рендер сторінки при не коректно введеному запиті.

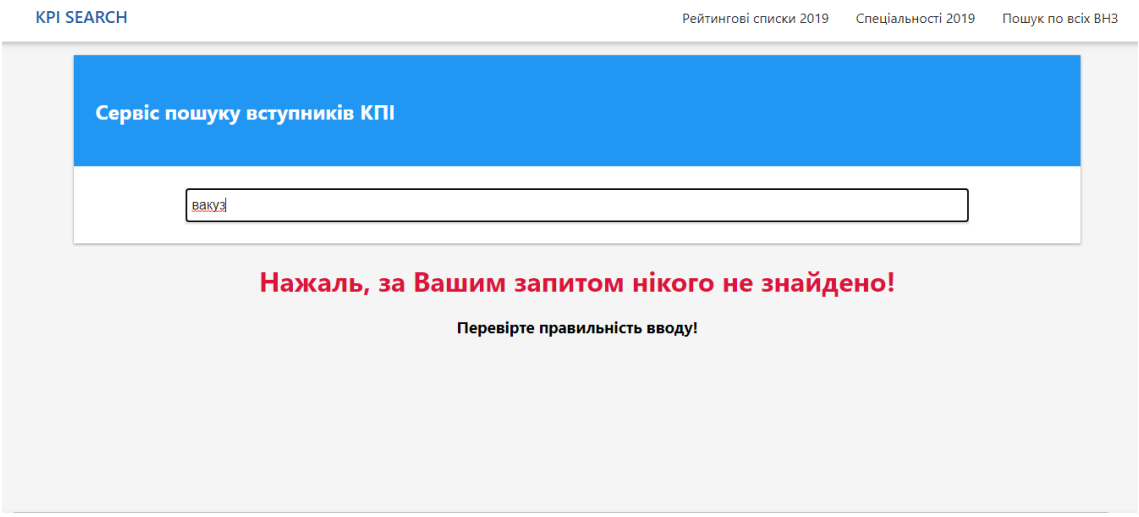

Рисунок 5.2.3 — Відображення помилки при не коректному введені

Шапка сайту містить модуль додаткових посилань для зручності користувача (Рисунок 5.2.7)

Рисунок 5.2.4 — Інтерфейс модулю додаткових посилань

Також, наша система дозволяє шукати студента:

- по рокам вступу, якщо вони повторюються в бд (Рисунок 5.2.5 і Рисунок 5.2.6)
- по унікальному ідентифікатору студента, якщо вони повторюють і ще й того самого року вступу (Рисунок 5.2.7)

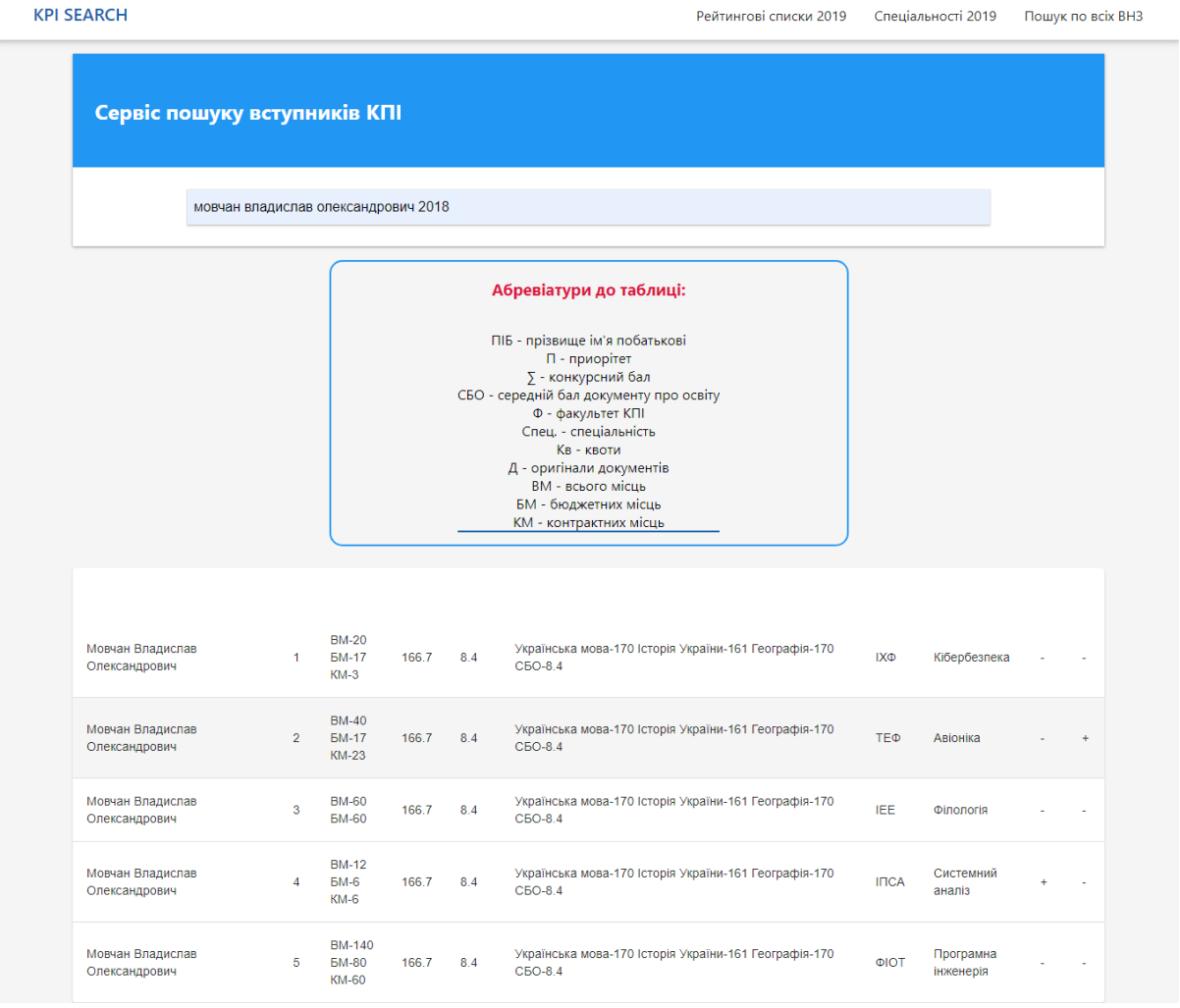

Рисунок 5.2.5 — Інформація про Мовчана Владислава Олександровича 2018 року вступу

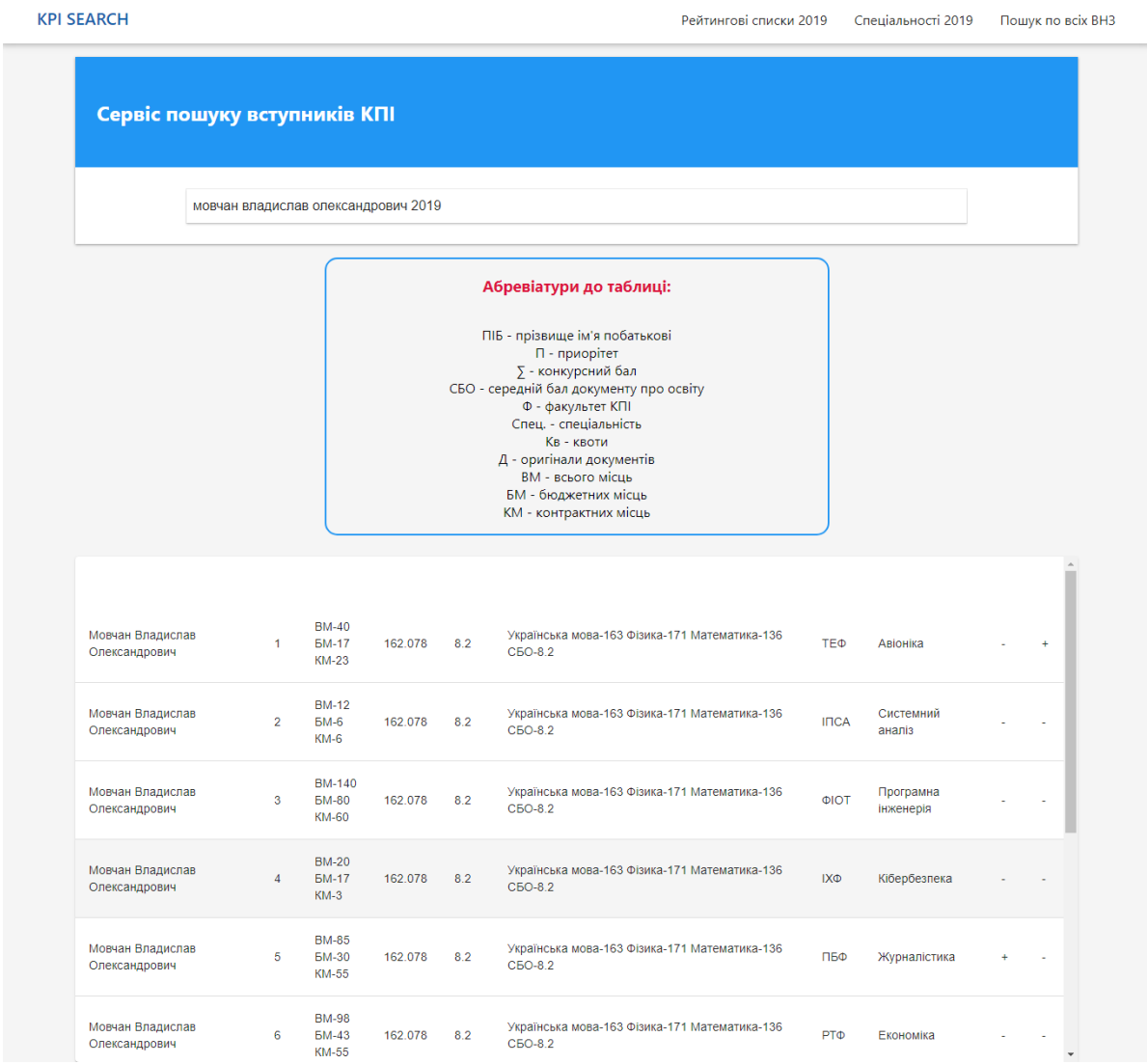

Рисунок 5.2.6 — Інформація про Мовчана Владислава Олександровича 2019 року вступу

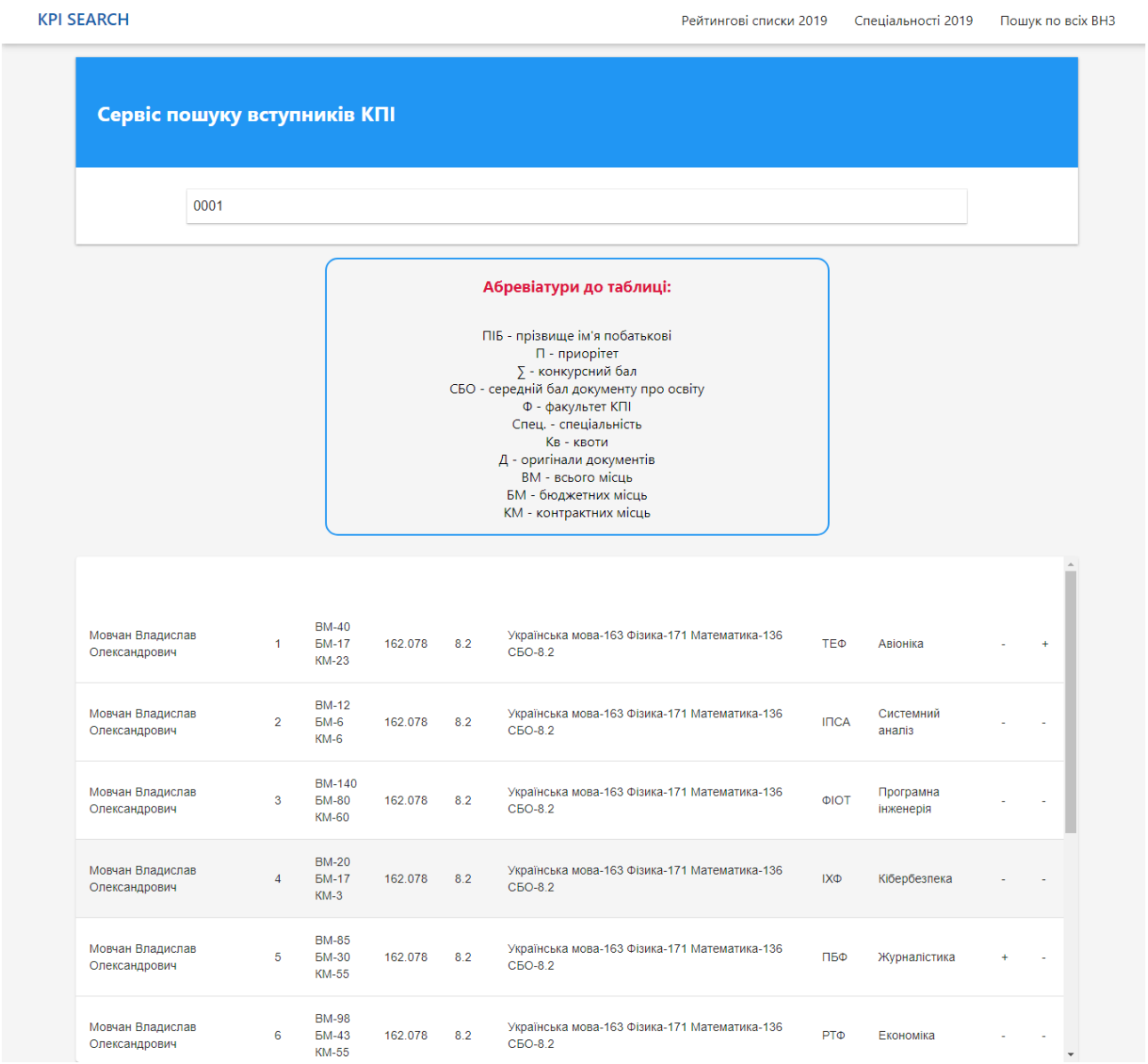

Рисунок 5.2.7 - Інформація про Мовчана Владислава Олександровича 2018 року вступу по ID

ВИСНОВКИ

У ході аналізу існуючих систем моніторингу було досліджено, що проблема ще не є вирішеною для сайту нашого вузу, хоча багато університетів вже мають таку систему.

Систему було написано на мові програмування JavaScript з використанням графічного веб-інтерфейсу.

Програму можна використовувати для дослідження інформації стосовно свого вступу, так і для дослідження вступу інших осіб з детальною інформацією відфільтровану відносно пріоритетів.

Також було проведено дослідження, щодо використання існуючих веб-технологій, таких як React, Redux, Redux-Saga, React-Router, MaterialUI, JavaScript, CSS4, HTML5, ES6, SCSS для розробки продукту, написано в середовищі WebStorm для максимально зручної взаємодії з користувачем.

Для роботи з даним сайтом необхідний лише комп’ютер низької потужності.

Побудований програмний продукт дозволяє моніторити в режимі реального часу вступників до КПІ. Користувач має змогу власноруч задавати параметри для пошуку в системі, щоб дослідити процес вступу обраної особи. Для нього виводяться таблиця з повною інформацією про кожну пошукову особу, така як: ПІБ абітурієнта, пріоритет заяви, місця, конкурсний бал, середній бал документа про освіту, складові конкурсного балу, коефіцієнти, факультет, спеціальність(напрям) / спеціалізація, вступ за квотою та оригінали документів.

Для побудованої моделі були проведені тестуванні, які підтвердили працездатність додатку.

Були досліджені бібліотеки, фреймворки, методи та алгоритми, на основі яких будувалась і навчалась модель. Проведено порівняння видів систем моніторингу і вибрано найбільш ефективний спосіб для створення власної.

Отже, ця робота покращила мої знання в новітніх технологіях для фронт-енд розробки та дала знання про предметну область систем моніторингу та вступної компанії. Було досліджено різноманітні підходи до розробки додатків та покращено знання у роботі веб-додатків. В результаті цього було створено систему моніторингу стану пріоритетів вступників до КПІ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Предметна область вступної компанії – Режим доступу: http://osvita.ua/
2. Концептуальна модель [Redux] – Режим доступу: https://redux.js.org/
3. Фраймворк для фронт-енд розробки на базі JS [React JS] – Режим доступу: https://ru.reactjs.org/
4. Аналог системи моніторингу для вступної компанії – Режим доступу: https://abit-poisk.org.ua/
5. Середовище розробки [WebStorm] – Режим доступу: https://www.jetbrains.com/ru-ru/webstorm/
6. Бібліотека для інтерфейсу таблиць [MaterialUI] – Режим доступу: https://material-ui.com/
7. Інструмент браузерної розробки [Redux dev tools] – Режим доступу: https://github.com/reduxjs/redux-devtools
8. Інструмент браузерної розробки [React dev tools] – Режим доступу: https://github.com/reduxjs/react-devtools
9. [Веб інтерфейс] – Режим доступу: https://www.compgramotnost.ru/internet-gramotnost/chto-takoe-web-interfejs-i-kak-im-vospolzovatsya
10. Архітектура проекту [MVC] – Режим доступу: https://web-creator.ru/articles/mvc
11. Територіальне розміщення університету[Електронний ресурс]. – Режим доступу: https://kpi.ua/location
12. Философский энциклопедический словарь. / Л. Ф. Ильичёв, П. Н. Федосеев, С. М. Ковалёв, В. Г. Панов.//Советская энциклопедия.– 1983.
13. Рефакторинг: улучшение существующего кода. / Фаулер М. // Пер. с англ. СПб: Символ-Плюс – 2003. – 432 с.

14. Исаев, Г. Н. Проектирование информационних систем: учебное пособие – М.: ОМЕГА-Л, 2015. – 424 с.
15. Соловьев, И.В. Проектирование информационних систем. Фундаментальний курс / И.В. Соловьев, А.А. Майоров; Моск. гос. ун-т геодезии и картографии . - М. : Акад. проект, 2015. - 399 с.

ДОДАТОК 1

Система моніторингу стану пріоритетів вступників до КПІ ім. Ігоря Сікорського

Текст програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ

Аркушів 10

Київ – 2020

Файл “index.js”:

```
import React from "react";
import Student from "../Students/Student/index";
import "./styles.css";

const Students = ({ info }) => {
  return (
    <div className="info">
      {info ? (
        <div>
          <Student info={info} />
        </div>
      ) : (
        <div className="error">
          <h1 className="sms">Нажаль, за Вашим запитом нікого не знайдено!</
h1>
          <h3>Перевірте правильність вводу!</h3>
        </div>
      )}
    </div>
  );
};

export default Students;
```

Файл “app.js”:

```
import React, { useEffect } from "react";
import { connect } from "react-redux";
import { getStudentsApi, getSearch } from "../../redux/students";
import Students from "../Students";
import "./styles.css";

const App = ({ getAPI, students, getSearchValue, input, match, history }) =>
 {
  useEffect(() => {
    getAPI();
  }, [students]);

  const changeInputHandler = (event) => {
    input = event.target.value.toLowerCase();
    history.push({
      pathname: match.params.category,
      search: input,
```

```
    });
  };

  const resultsBySearch =
    students.length !== 0
      ? students.filter((data) => {
          const names = data.names.toLowerCase();
          const names2 = decodeURI(
            history.location.search.slice(1).toLowerCase()
          );
          return ~names.indexOf(names2);
        })
      : null;

  const priority = resultsBySearch
    ? resultsBySearch.map((el) =>
        el.info.sort((a, b) => (a.priority > b.priority ? 1 : -1))
      )
    : null;

  return (
    <div className="wrapper">
      <header className="header">
        <a href="https://kpi.ua/" className="logo">
          kpi search
        </a>
        <nav>
          <ul className="header-navigation">
            <li className="list">
              <a href="https://abit-poisk.org.ua/rate-
review/" className="link">
                Рейтингові списки 2019
              </a>
            </li>
            <li className="list">
              <a
                href="https://abit-poisk.org.ua/specialities2019/"
                className="link"
              >
                Спеціальності 2019
              </a>
            </li>
            <li className="list">
              <a href="https://abit-poisk.org.ua/#" className="link">
```

```
                Пошук по всіх ВНЗ
              </a>
            </li>
          </ul>
        </nav>
      </header>
      <div className="main-content">
        <div className="card">
          <div className="card-header">
            <h3 className="card-header-
title">Сервіс пошуку вступників КПІ</h3>
          </div>
          <div className="card-content">
            <div className="input-box">
              <input
                type="text"
                id="searchInput"
                placeholder="ПІБ (рік вступу) або id студента(в форматі 0123
4)"
                className="search-input"
                onChange={() => {
                  getSearchValue(decodeURI(input));
                }}
                onInput={changeInputHandler}
                value={decodeURI(history.location.search.slice(1))}
              />
            </div>
          </div>
        </div>
      </div>
      <div className="info">
        {priority && history.location.search.slice(1) ? (
          <Students info={priority[0]} />
        ) : null}
      </div>
    </div>
  );
};

const mapStateToProps = (state) => ({
  students: state.studentsList.students,
  input: state.studentsList.input,
});
```

```
const mapDispatchToProps = (dispatch) => ({
  getAPI: () => dispatch(getStudentsApi()),
  getSearchValue(value) {
    dispatch(getSearch(value));
  },
});

export default connect(mapStateToProps, mapDispatchToProps)(App);
```

Файл “students.js”:

```
import React from "react";
import Student from "../Students/Student/index";
import "./styles.css";

const Students = ({ info }) => {
  return (
    <div className="info">
      {info ? (
        <div>
          <Student info={info} />
        </div>
      ) : (
        <div className="error">
          <h1 className="sms">Нажаль, за Вашим запитом нікого не знайдено!</
h1>
          <h3>Перевірте правильність вводу!</h3>
        </div>
      )}
    </div>
  );
};

export default Students;
```

Файл “student.js”:

```
import React from "react";
import { makeStyles } from "@material-ui/core/styles";
import Paper from "@material-ui/core/Paper";
import Table from "@material-ui/core/Table";
import TableBody from "@material-ui/core/TableBody";
import TableCell from "@material-ui/core/TableCell";
```

```
import TableContainer from "@material-ui/core/TableContainer";
import TableHead from "@material-ui/core/TableHead";
import TableRow from "@material-ui/core/TableRow";

import "./styles.css";

const columns = [
  { id: "name", label: "ПІБ", width: "20%" },
  { id: "priority", label: "П" },
  { id: "places", label: "Місця", width: "5%" },
  { id: "concurs", label: "∑" },
  { id: "sbo", label: "СБО" },
  { id: "kof", label: "Складові конкурсного балу", width: "35%" },
  { id: "fac", label: "Ф" },
  { id: "speciality", label: "Спец." },
  { id: "kvotu", label: "Кв" },
  { id: "originDocs", label: "Д" },
];

const useStyles = makeStyles({
  root: {
    width: "100%",
  },
  container: {
    maxHeight: 600,
  },
});

export default function Student({ info }) {
  const classes = useStyles();

  return (
    <div className="student_info">
      <div className="help">
        <h3 className="zag">Абревіатури до таблиці:</h3>
        <ul className="list">
          <li>ПІБ - прізвище ім'я побатькові</li>
          <li>П - приорітет</li>
          <li>∑ - конкурсний бал</li>
          <li>СБО - середній бал документу про освіту</li>
          <li>Ф - факультет КПІ</li>
          <li>Спец. - спеціальність</li>
          <li>Кв - квоти</li>
          <li>Д - оригінали документів</li>
```

```
    <li>ВМ - всього місць</li>
    <li>БМ - бюджетних місць</li>
    <li>КМ - контрактних місць</li>
  </ul>
</div>
<Paper className={classes.root}>
  <TableContainer className={classes.container}>
    <Table stickyHeader aria-label="sticky table">
      <TableHead>
        <TableRow>
          {columns.map((column) => (
            <TableCell key={column.id}>{column.label}</TableCell>
          ))}
        </TableRow>
      </TableHead>
      <TableBody>
        {info
          ? info.map((row) => {
              return (
                <TableRow
                  hover
                  role="checkbox"
                  tabIndex={-1}
                  key={row.priority}
                >
                  {columns.map((column) => {
                    const value = row[column.id];
                    return (
                      <TableCell
                        key={column.id}
                        style={{ width: column.width }}
                      >
                        {column.format && typeof value === "number"
                          ? column.format(value)
                          : value}
                      </TableCell>
                    );
                  })}
                </TableRow>
              );
            })
          : null}
      </TableBody>
    </Table>
```

```
        </TableContainer>
      </Paper>
    </div>
  );
}
```

Файл “store.js”:

```
// Library
import { createStore, combineReducers, applyMiddleware } from "redux";
import createSagaMiddleware from "redux-saga";
// Components
import reducer, { parsing } from "./students";

// STORE
const sagaMiddleware = createSagaMiddleware();
let store = createStore(
  combineReducers({
    studentsList: reducer,
  }),
  applyMiddleware(sagaMiddleware)
);

sagaMiddleware.run(parsing);

export default store;
```

Файл “redux.js”:

```
import { takeLatest } from "redux-saga/effects";
import { put, call } from "redux-saga/effects";

// Actions
export const GET_STUDENTS = "GET_STUDENTS";
export const GET_STUDENTS_API = "GET_STUDENTS_API";
export const GET_SEARCH = "GET_SEARCH";

// Default State
export const initialState = {
  students: [],
  search: [],
  input: "",
};

// Dispatch
export const getStudentsApi = () => ({ type: GET_STUDENTS_API });
export const getSearch = (value) => ({ value, type: GET_SEARCH });

// Reducer
const reducer = (state = initialState, action) => {
  switch (action.type) {
    case GET_STUDENTS:
      return {
        ...state,
        students: action.data,
      };
    case GET_SEARCH:
      return {
        ...state,
        input: action.value,
      };
    default:
      return state;
  }
};
```

```
// REDUX-SAGA
export function* getStudents() {
  const url = "data.json";
  try {
    const data = yield call(() => {
      return fetch(url).then((res) => res.json());
    });
    yield put({ type: GET_STUDENTS, data });
  } catch (error) {
    console.error(error);
  }
}

export function* parsing() {
  yield takeLatest(GET_STUDENTS_API, getStudents);
}

export default reducer;
```

ДОДАТОК 2

Система моніторингу стану пріоритетів вступників до КПІ ім. Ігоря Сікорського

Опис програми

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТМ

Аркушів 8

Київ – 2020

АНОТАЦІЯ

Додаток надає персоналізовані рекомендації користувачу.

Побудований програмний продукт дозволяє моніторити в режимі реального часу вступників до КПІ. Користувач має змогу власноруч задавати параметри для пошуку в системі, щоб дослідити процес вступу обраної особи. Для нього виводяться таблиця з повною інформацією про кожну пошукову особу, така як: ПІБ абітурієнта, пріоритет заяви, місця, конкурсний бал, середній бал документа про освіту, складові конкурсного балу, коефіцієнти, факультет, спеціальність(напрям) / спеціалізація, вступ за квотою та оригінали документів.

Для розв’язання поставлених задач, використовувались мови програмування HTML/CSS, Javascript, також використовувались алгоритми для сортування студентів по пріоритетності. Для мокової бази даних було додано асинхронних запит на сервер за допомогою бібліотеки Redux-Saga та імітація самої бд за допомогою бібліотеки Redux у форматі JSON. Система була написана на фреймворці React. Розробка системуздійснюється в середовищі WebStorm 2020.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Відповідно до назви дипломної роботи, розроблений продукт дозволяє в реальному часі здійснювати моніторинг пріоритету станів вступників до КПІ ім. Ігоря Сікорського.

Користувач, для роботи з системою, повинен мати комп’ютер або мобільний телефон.

Для використання додатку потрібно запустити веб браузер та перейти за посиланням.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Розроблений програмний засіб був створений, щоб автоматизувати деякі процеси для абітурієнтів та створити систему, яка б дозволяла в режимі реального часу здійснювати моніторинг стану пріоритетів для студентів Київського Політехнічного Інституту ім. Ігоря Сікорського.

Основний функціонал системи – це моніторинг вступної компанії конкретного студента в режимі реального часу на основі аналізу його пріоритетів, та виведення інформації про кожного в новий мінімалістичний інтерфейс.

Сайт відобразить користувачу всю відфільтровану по пріоритетам інформацію стосовно пошукової особи пов’язану із вступом до КПІ. Аналізуючи ці дані, можна передбачити багато факторів, таких як:

- ПІБ абітурієнта
- Пріоритет заяви
- Місця
- Конкурсний бал
- Середній бал документа про освіту
- Складові конкурсного балу, коефіцієнти
- Факультет
- Спеціальність(напрям) / Спеціалізація
- Вступ за квотою
- Подано оригінали документів

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Загальний принцип роботи додатку:

1) Користувач заходить на веб-сайт та може переглядати модуль додаткових посилань, для зручності, щоб відразу перейти на сайт КПІ або аналогічних систем моніторингу чи інших корисних посилань для споживача.
2) Для того, щоб оцінювати свою вступну компанію, чи своїх конкурентів, користувач має змогу скористатися модулем живого пошуку і ввести запит на пошукову особу.
3) Після цього система отримує дані від користувача та відповідно їх впорядковує з сортуванням по пріоритетності і виводить інформацію щодо конкретного студента та його заявок до вступу.
4) Користувач може аналізувати та порівнювати інформацію про свій вступ з іншими студентами.

-7-

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Дана система розроблювалась за допомогою мови програмування JavaScript. Для розробки використовувались фреймворк React.js. Для реалізації моделі використовувались технології Redux, Redux-Saga. База даних отримується в форматі JSON.

-8-

ВХІДНІ І ВИХІДНІ ДАНІ

Вхідними даними є:

— база даних студентів;

— пошуковий запит;

Вихідними даними є:

— абревіатури до таблиць;

— інформацію про пошукову особу (ПІБ абітурієнта, пріоритет заяви, місця, конкурсний бал, середній бал документа про освіту, складові конкурсного балу, коефіцієнти, факультет, спеціальність(напрям)/спеціалізація, вступ за квотою, подані оригінали документів);